



PATENT ABSTRACTS OF JAPAN

(11) Publication number: **06274384 A**

(43) Date of publication of application: **30.09.94**

(51) Int. Cl

G06F 12/00

(21) Application number: 05062181

(71) Applicant: **N T T DATA TSUSHIN KK**

(22) Date of filing: 22.03.93

(72) Inventor: **TANIMURA MORIMASA**

(54) **EXTRACTING/UPDATING DEVICE OF
EXECUTION FILE DIFFERENCE AND
EXTRACTING METHOD FOR EXECUTION FILE
DIFFERENCE**

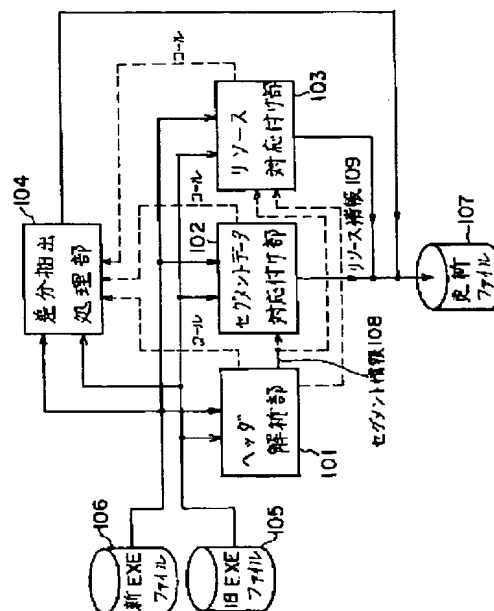
101 to 103 and saves the difference in the update file 107.

COPYRIGHT: (C)1994,JPO&Japio

(57) Abstract:

PURPOSE: To reduce the size of an update file storing difference information to the old execution file of a new execution file.

CONSTITUTION: Concerning respective old and new EXE files 105 and 106, a header analysis part 101 extracts the areas of their headers and the areas of respective tables and calls a difference extraction processing part 104 concerning these areas. A segment data coordination part 102 coordinates segment data between the files 105 and 106 and calls the difference extraction processing part 104 concerning the coordinated area. A resource coordination part 103 coordinates a resource between the files 105 and 106 and calls the difference extraction processing part 104 concerning the coordinated area. A difference extraction processing part 104 extracts the difference of a byte unit to the file 105 of the file 106 from the top and the end of the areas specified from the respective parts



(19)日本国特許庁(JP)

(12) 公開特許公報(A)

(11)特許出願公開番号

特開平6-274384

(43)公開日 平成6年(1994)9月30日

(51)Int.Cl.⁵

G 0 6 F 12/00

識別記号

5 1 0 B

庁内整理番号

8944-5B

F I

技術表示箇所

審査請求 未請求 請求項の数 6 O L (全 30 頁)

(21)出願番号 特願平5-62181

(22)出願日 平成5年(1993)3月22日

(71)出願人 000102728

エヌ・ティ・ティ・データ通信株式会社
東京都江東区豊洲三丁目3番3号

(72)発明者 谷村 守正

東京都江東区豊洲三丁目3番3号 エヌ・
ティ・ティ・データ通信株式会社内

(74)代理人 弁理士 大菅 義之

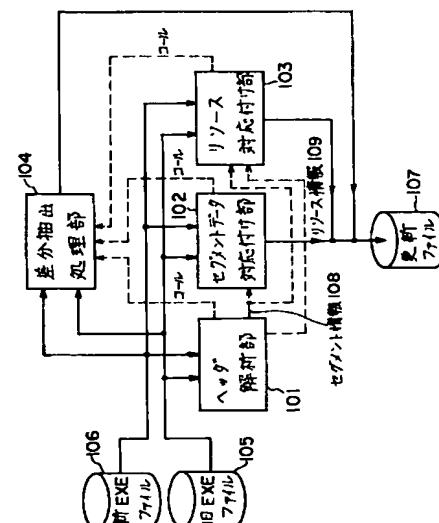
(54)【発明の名称】 実行ファイル差分抽出／更新装置及び実行ファイル差分抽出方法

(57)【要約】

【目的】 本発明は、新実行ファイルの旧実行ファイルに対する差分情報を格納する更新ファイルのサイズを削減することを目的とする。

【構成】 ヘッダ解析部101は、旧EXEファイル105と新EXEファイル106のそれぞれにつき、そのヘッダの領域及び各テーブル部の領域を抽出し、それらの領域につき差分抽出処理部104をコールする。セグメントデータ対応付け部102は、ファイル105と106との間のセグメントデータの対応付けを行い、対応付けされた領域について差分抽出処理部104をコールする。リソース対応付け部103は、ファイル105と106との間のリソースの対応付けを行い、対応付けされた領域について差分抽出処理部104をコールする。差分抽出処理部104は、101、102、又は103の各部から指定された領域の先頭から末尾まで、ファイル106の105に対するバイト単位の差分を抽出し、その差分を更新ファイル107にセーブする。

本発明の実施例の構成図



【特許請求の範囲】

【請求項1】 旧実行ファイルを新実行ファイルに更新するための更新ファイルを、新実行ファイルの旧実行ファイルに対する差分として抽出する実行ファイル差分抽出装置において、

前記旧実行ファイル及び前記新実行ファイルのそれぞれを、所定の論理単位毎に複数の領域に分割する領域分割手段と、

該領域分割手段によって分割された前記旧実行ファイルの複数の領域と前記新実行ファイルの複数の領域との間で、互いに対応する領域同士を対応付けする領域対応付け手段と、

該領域対応付け手段によって対応付けされた前記旧実行ファイルの領域と前記新実行ファイルの領域との間で、前記旧実行ファイルの領域のデータに対する前記新実行ファイルの領域のデータの差分情報を抽出する差分抽出処理手段と、

該差分抽出処理手段において抽出された差分情報を前記更新ファイルに格納すると共に、前記領域対応付け手段において対応付けが行われなかった前記旧実行ファイルの領域又は前記新実行ファイルの領域に対応する差分情報を抽出しそれを前記更新ファイルに格納する更新ファイル生成手段と、

を有することを特徴とする実行ファイル差分抽出装置。

【請求項2】 新実行ファイルの旧実行ファイルに対する差分が抽出されて更新ファイルとされ、旧実行ファイルと更新ファイルとから新実行ファイルが生成される実行ファイル更新装置において、

請求項1に記載の更新ファイル生成手段により生成された更新ファイルから前記差分情報を順次取り出し、該差分情報と前記旧実行ファイルとから、前記新実行ファイルを生成するファイル更新手段を有する、ことを特徴とする実行ファイル更新装置。

【請求項3】 新実行ファイルの旧実行ファイルに対する差分が抽出されて更新ファイルとされ、旧実行ファイルと更新ファイルとから新実行ファイルが生成される実行ファイル差分抽出／更新装置において、

前記旧実行ファイル及び前記新実行ファイルのそれぞれを、所定の論理単位毎に複数の領域に分割する領域分割手段と、

該領域分割手段によって分割された前記旧実行ファイルの複数の領域と前記新実行ファイルの複数の領域との間で、互いに対応する領域同士を対応付けする領域対応付け手段と、

該領域対応付け手段によって対応付けされた前記旧実行ファイルの領域と前記新実行ファイルの領域との間で、前記旧実行ファイルの領域のデータに対する前記新実行ファイルの領域のデータの差分情報を抽出する差分抽出処理手段と、

該差分抽出処理手段において抽出された差分情報を前記

更新ファイルに格納すると共に、前記領域対応付け手段において対応付けが行われなかった前記旧実行ファイルの領域又は前記新実行ファイルの領域に対応する差分情報を抽出しそれを前記更新ファイルに格納する更新ファイル生成手段と、

該更新ファイル生成手段により生成された更新ファイルから前記差分情報を順次取り出し、該差分情報と前記旧実行ファイルとから、前記新実行ファイルを生成するファイル更新手段と、

を有することを特徴とする実行ファイル差分抽出／更新装置。

【請求項4】 旧実行ファイルを新実行ファイルに更新するための更新ファイルを、新実行ファイルの旧実行ファイルに対する差分として抽出する実行ファイル差分抽出方法において、

前記旧実行ファイル及び前記新実行ファイルのそれぞれを、所定の論理単位毎に複数の領域に分割し、該領域の分割によって分割された前記旧実行ファイルの複数の領域と前記新実行ファイルの複数の領域との間で、互いに対応する領域同士を対応付けし、

該領域の対応付けによって対応付けされた前記旧実行ファイルの領域と前記新実行ファイルの領域との間で、前記旧実行ファイルの領域のデータに対する前記新実行ファイルの領域のデータの差分情報を抽出し、該抽出された差分情報を前記更新ファイルに格納すると共に、前記領域の対応付けにおいて対応付けが行われなかった前記旧実行ファイルの領域又は前記新実行ファイルの領域に対応する差分情報を抽出しそれを前記更新ファイルに格納する、

ことを特徴とする実行ファイル差分抽出方法。

【請求項5】 前記論理単位は、前記旧実行ファイル又は前記新実行ファイルのデータ属性の単位であるセグメント、リソース、ヘッダ、又はテーブルであることを特徴とする請求項1乃至4の何れか1項に記載の実行ファイル差分抽出／更新装置又は実行ファイル差分抽出方法。

【請求項6】 前記差分情報は、前記新実行ファイルにおいて前記旧実行ファイルに対してデータの挿入、削除、又は置換が行われる旨の属性情報、その対象となるデータの長さ情報、及びその対象となるデータとから構成される、

ことを特徴とする請求項1乃至5の何れか1項に記載の実行ファイル差分抽出／更新装置又は実行ファイル差分抽出方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 本発明は、バグの修正又は部分的な機能拡張を目的として実行ファイルが更新される場合に、新実行ファイルの旧実行ファイルに対する差分が抽出されて更新ファイルとされ、旧実行ファイルと更新フ

10

20

30

40

50

ファイルとから新実行ファイルが生成される実行ファイル差分抽出／更新装置及び方法に関する。

【0002】

【従来の技術】ユーザに実行ファイルが配付されその実行ファイル（EXEファイル）により所定サービスが提供されている場合、その実行ファイルに存在するバグを修正したり、その実行ファイルによって実現される機能の部分的な拡張を目的として、図28(a)に示されるように、ユーザに配付されている実行ファイルにおいて、新EXEファイルから旧EXEファイルへの更新がなされる場合がある。

【0003】一方、最近では、コンピュータシステムの機能の発達により、実行ファイルのプログラムサイズが数百キロバイトから数メガバイト或いは数十メガバイトに達しつつある。従って、上述のような実行ファイルの更新時に、新EXEファイルがそのまま配付されるのは、配付媒体に多くのデータ容量が要求されることになり、不経済である。

【0004】例えば、配付媒体が図29に示されるようなネットワークである場合、センタTCのホストコンピュータHOSTは、回線を介して、事務所TC内のホストコンピュータHOSTに対して新EXEファイルを配信することになるが、新EXEファイルが上述のような大きなサイズを有する場合、回線使用料が高くついてしまい、不経済である。

【0005】このような問題に対し、従来、新EXEファイルそのものが配付されるのではなく、図28(b)に示されるように、旧EXEファイルと新EXEファイルとの間で比較処理がなされ、新EXEファイルの旧EXEファイルに対する差分が抽出されて更新ファイルとされ、その更新ファイルがユーザに配付されるという形態が採られるようになってきている。この場合、ユーザ側では、図28(b)に示されるように、所定の更新プログラムによる更新処理によって、旧EXEファイルと更新ファイルとから新EXEファイルが生成される。新EXEファイルと旧EXEファイルとでは、プログラムコードが同じ部分が存在する場合が多いために、それらの差分である更新ファイルのサイズは、新EXEファイルのサイズに比較して小さい場合が多い。従って、更新ファイルがユーザに配付されることにより、配付媒体に要求されるデータ容量を削減することができる。

【0006】図29の例においては、センタTCのホストコンピュータHOSTは、回線を介して、事務所TC内のホストコンピュータHOSTに対して更新ファイルを配信する。事務所TC内のホストコンピュータHOSTでは、センタ間通信部が、上記更新ファイルを受信した後、更新プログラムを起動することにより、旧EXEファイルと更新ファイルとから新EXEファイルを生成する。この新EXEファイルは、ファイル操作部を介して正ディスク及び副ディスクに保存される。そして、事

務所TCのホストコンピュータHOST内の配信処理管理部が、新EXEファイルを、事務所内通信部からLAN（ローカルエリアネットワーク）を介して、クライアントサーバCSVに配信する。クライアントサーバCSVでは、事務所内通信部が上記新EXEファイルを受信する。この受信された新EXEファイルは、ファイル操作部を介して正ディスク及び副ディスクに保存される。

【0007】

【発明が解決しようとする課題】しかし、上述のような従来の差分抽出方式では、旧EXEファイルの先頭及び新EXEファイルの先頭から各ファイルの末尾までのファイル全体にわたって、バイト単位でデータが一致するか否かが判定され、一致しないデータが差分のデータとして抽出されている。このために、例えば新EXEファイルの途中で1バイトでもデータの削除又は挿入が行われていると、そのバイト以降では全て旧EXEファイルのデータと新EXEファイルのデータとが一致しなくなり、その結果、更新ファイルのサイズが大きくなってしまふという問題点を有している。従って、図29の例のようにネットワークを介して更新ファイルが配信される場合などで、新EXEファイルがそのまま配信される場合に比較してそれほど回線の使用効率が向上しないという結果になってしまう。

【0008】本発明は、更新ファイルのサイズを削減することを目的とする。

【0009】

【課題を解決するための手段】本発明は、まず、実行ファイル差分抽出装置として、以下の構成を有する。領域分割手段は、旧実行ファイル及び新実行ファイルのそれぞれを、所定の論理単位毎に複数の領域に分割する。この論理単位は、例えば、旧実行ファイル又は新実行ファイルのデータ属性の単位であるセグメント、リソース、ヘッダ、又はテーブルである。

【0010】次に、領域対応付け手段は、領域分割手段によって分割された旧実行ファイルの複数の領域と新実行ファイルの複数の領域との間で、互いに対応する領域同士を対応付けする。

【0011】また、差分抽出処理手段は、領域対応付け手段によって対応付けされた旧実行ファイルの領域と新実行ファイルの領域との間で、旧実行ファイルの領域のデータに対する新実行ファイルの領域のデータの差分情報を抽出する。この差分情報は、例えば、新実行ファイルにおいて旧実行ファイルに対してデータの挿入、削除、又は置換が行われる旨の属性情報、その対象となるデータの長さ情報、及びその対象となるデータとから構成される。

【0012】そして、更新ファイル生成手段は、差分抽出処理手段において抽出された差分情報を更新ファイルに格納すると共に、領域対応付け手段において対応付けが行われなかった旧実行ファイルの領域又は新実行ファ

10

20

30

40

50

イルの領域に対応する差分情報を抽出しそれを更新ファイルに格納する。

【0013】また、本発明は、実行ファイル更新装置として、上述の更新ファイル生成手段により生成された更新ファイルから前記差分情報を順次取り出し、その差分情報と旧実行ファイルとから、新実行ファイルを生成するファイル更新手段を有するように構成される。

【0014】

【作用】本発明においては、差分抽出処理が実行される場合、例えばOS/2システムにおける実行ファイルでは、DOS EXE 形式の領域、OS/2 EXEヘッダ領域、各テーブル領域、セグメントデータ領域、及びリソース領域のそれぞれの論理単位毎に、対応付けが行われた上で差分抽出処理が実行される。

【0015】このため、旧実行ファイルと新実行ファイルとの間で、各領域のデータが一致する確率が高くなり、その結果、生成される更新ファイルのサイズが大幅に削減される。

【0016】

【実施例】以下、図面を参照しながら本発明の実施例につき詳細に説明する。図1は、本発明の実施例の構成図である。

【0017】まず、差分抽出処理部104は、後述するヘッダ解析部101、セグメントデータ対応付け部102、又はリソース対応付け部103から指定された領域の先頭から末尾まで、新EXEファイル106の旧EXEファイル105に対するバイト単位の差分を抽出し、その差分を更新ファイル107にセーブする。

【0018】ヘッダ解析部101は、旧EXEファイル105と新EXEファイル106のそれぞれについて、そのDOS EXE ヘッダの領域及びOS/2 EXEヘッダの領域を抽出し、それらの領域について差分抽出処理部104に対して差分抽出処理を指示する。また、ヘッダ解析部101は、旧EXEファイル105と新EXEファイル106のそれぞれについて、上述の抽出されたDOS EXE ヘッダ及びOS/2 EXEヘッダの内容を解析することにより後述する各テーブル部の領域を抽出し、抽出された各テーブル部領域についても差分抽出処理部104に対して差分抽出処理を指示する。更に、ヘッダ解析部101は、旧EXEファイル105と新EXEファイル106のそれぞれについて、抽出されたセグメントテーブル部から後述するセグメント情報108を抽出して、それを後述するセグメントデータ対応付け部102及びリソース対応付け部103に設定し、また、抽出されたリソーステーブル部から後述するリソース情報109を抽出して、それを後述するリソース対応付け部103に設定する。

【0019】セグメントデータ対応付け部102は、ヘッダ解析部101からのセグメント情報108に基づいて、旧EXEファイル105と新EXEファイル106との間のセグメントデータの対応付けを行い、対応付け

された各セグメントデータのペアの領域について差分抽出処理部104に対して差分抽出処理を指示する。旧EXEファイル105に存在するセグメントデータに対応するものが新EXEファイル106に存在しない場合は、そのセグメントデータを削除すべき旨の差分情報を更新ファイル107にセーブする。逆に、新EXEファイル106に存在するセグメントデータに対応するものが旧EXEファイル105に存在しない場合は、そのセグメントデータを挿入すべき旨の差分情報を更新ファイル107にセーブする。

【0020】リソース対応付け部103は、ヘッダ解析部101からのセグメント情報108及びリソース情報109に基づいて、旧EXEファイル105と新EXEファイル106との間のリソースの対応付けを行い、対応付けされた各リソースのペアの領域について差分抽出処理部104に対して差分抽出処理を指示する。旧EXEファイル105に存在するリソースに対応するものが新EXEファイル106に存在しない場合は、そのリソースを削除すべき旨の差分情報を更新ファイル107にセーブする。逆に、新EXEファイル106に存在するリソースに対応するものが旧EXEファイル105に存在しない場合は、そのリソースを挿入すべき旨の差分情報を更新ファイル107にセーブする。

【0021】上述の構成を有する実施例の動作について、以下に順次説明する。まず、図2は、旧EXEファイル105及び新EXEファイル106の共通のファイル形式を示した図である。

【0022】本実施例で対象とされるファイルは、コンピュータのためのマルチタスクオペレーティングシステムの1つであるOS/2システムのもとで実行可能なEXEファイルである。

【0023】このEXEファイルは、コンピュータのための従来からあるシングルタスクオペレーティングシステムの1つであるMS-DOSシステムのもとでも実行可能のように構築されている。そのため、このEXEファイルは、図2に示されるように、DOS EXE 形式の領域とOS/2 EXE形式の領域とから構成されている。

【0024】まず、ファイルの前半部分には、DOS EXE 形式の領域が存在する。DOS EXE 形式の領域は、MS-DOS 用プログラムを実行するための各種情報と後述するOS/2 EXEヘッダへのポインタ情報等が格納されるDOS EXE ヘッダ領域と、MS-DOSシステムのもとで動作するプログラムの本体が格納されるDOS ロードモジュール領域とから構成される。

【0025】次に、ファイルの後半部分には、OS/2 EXE 形式の領域が存在する。OS/2 EXE形式の領域は、OS/2用プログラムを実行するための各種情報と後述する各種テーブル部へのポインタ情報等が格納されるOS/2 EXEヘッダ領域、後述する各種テーブル部領域、OS/2システムのもとで動作するプログラムの本体が格納されるセグメン

10

20

30

40

50

トデータ領域、及びOS/2プログラムで利用されるリソースと呼ばれるデータが格納されるリソース領域とから構成される。

【0026】本実施例においては、差分抽出処理が実行される場合、DOS EXE 形式の領域、OS/2 EXEヘッダ領域、及び各テーブル部領域のそれぞれの領域毎に、旧EXEファイル105と新EXEファイル106との差分抽出処理が実行され、その後、旧EXEファイル105と新EXEファイル106との間のセグメントデータ及びリソースの対応付けが行われた上で、その対応付けされた各セグメントデータ及びリソースのペアの領域毎に差分抽出処理が実行されることが、大きな特徴である。このように、所定の論理単位毎に差分抽出処理が実行されることにより、従来技術に比較して、生成される更新ファイル107のサイズを大幅に削減することが可能となる。

【0027】次に、ヘッダ解析部101、セグメントデータ対応付け部102、リソース対応付け部103、及び差分抽出処理部104の各部の動作について、順次説明する。

【0028】本実施例では、まず、ヘッダ解析部101が起動される。ヘッダ解析部101で実行される処理の動作フローチャートは、図3及び図4に示される。このヘッダ解析部101の動作は、特に図示しないコンピュータシステムにおいて、CPU（中央処理装置）が、ディスク装置などに記憶された旧EXEファイル105及び新EXEファイル106に対して、メモリに記憶された制御プログラムを実行する動作の一部として実現される。

【0029】まず、ステップS301で、旧EXEファイル105及び新EXEファイル106のそれぞれについて、DOS EXE ヘッダ内のファイルオフセット0に格納されている1ワードのコードが“MZ”であるか否かが判定される（図5参照）。そのコードが“MZ”でない場合、そのファイルはEXEファイルではないため、ステップS323でエラー表示が行われ差分抽出処理を終了する。

【0030】上述のコードが“MZ”である場合、ステップS302で、旧EXEファイル105及び新EXEファイル106のそれぞれについて、DOS EXE ヘッダ内のファイルオフセット18h（“h”は、それが付加された1ワードの値が16進数であることを示す。）に格納されている値が40hであるか否かが判定される（図5参照）。その値が40hでない場合は、そのファイルはOS/2 EXE形式のファイルではないため、ステップS323でエラー表示が行われ差分抽出処理を終了する。

【0031】上述の値が40hである場合は、ステップS303で、旧EXEファイル105及び新EXEファイル106のそれぞれについて、DOS EXE ヘッダ内のファイルオフセット3Chに格納されている値で示される

1ワードのオフセットアドレスがOS/2 EXEの領域のアドレスであるか否かが判定される（図5参照）。そのアドレスがOS/2 EXEの領域のアドレスでない場合は、そのファイルはOS/2 EXE形式のファイルではないため、ステップS323でエラー表示が行われ差分抽出処理を終了する。

【0032】上述のアドレスがOS/2 EXEの領域のアドレスである場合は、更にステップS304で、旧EXEファイル105及び新EXEファイル106のそれぞれについて、OS/2 EXEヘッダ内のOS/2 EXE相対オフセット（OS/2 EXE形式の領域の先頭のアドレスが0とされる場合の相対オフセット）の36hに格納されている1ワードの値が1であるか否かが判定される（図5参照）。その値が1でない場合は、そのファイルはOS/2 EXE形式のファイルではないため、ステップS323でエラー表示が行われ差分抽出処理を終了する。

【0033】上述の値が1である場合、ステップS305で、旧EXEファイル105と新EXEファイル106のそれぞれにつき、DOS EXE 形式のデータ領域が、ファイルオフセット0からDOS EXE ヘッダ内のファイルオフセット3Chの内容として示されるOS/2 EXEの領域のアドレスの手前のアドレスまでの領域として抽出される（図2参照）。

【0034】次に、ヘッダ解析部101は、ステップS306で、差分抽出処理部104をコールすると共に、旧EXEファイル105と新EXEファイル106のそれぞれにつき抽出されたDOS EXE 形式のデータ領域を差分抽出処理部104に渡す。この結果、差分抽出処理部104は、後述する差分抽出処理により、ヘッダ解析部101から渡されたDOS EXE 形式のデータ領域の先頭から末尾まで、新EXEファイル106の旧EXEファイル105に対するバイト単位の差分を抽出し、その差分を更新ファイル107にセーブする。

【0035】次に、ステップS307で、旧EXEファイル105及び新EXEファイル106における図5に示されるOS/2 EXEヘッダの内容のうち、●印が付与されたデータが抽出され、それぞれ図6に示される各変数に格納される。この変数は、特に図示しないメモリに保持される。これらの変数については後述する。

【0036】続いて、ステップS308で、旧EXEファイル105と新EXEファイル106のそれぞれについて、OS/2 EXEヘッダ領域が、前述したDOS EXE ヘッダ内のファイルオフセット3Chの内容として示されるOS/2 EXEの領域のアドレスから変数segtbl off（図6参照）で示されるセグメントテーブル部の先頭アドレスの手前のアドレスまでの領域として抽出される（図2参照）。

【0037】次に、ヘッダ解析部101は、ステップS309で、差分抽出処理部104をコールすると共に、旧EXEファイル105と新EXEファイル106のそ

10

20

30

40

50

れぞれについて抽出されたOS/2 EXEヘッダ領域を差分抽出処理部104に渡す。この結果、差分抽出処理部104は、後述する差分抽出処理により、ヘッダ解析部101から渡されたOS/2 EXEヘッダ領域の先頭から末尾まで、新EXEファイル106の旧EXEファイル105に対するバイト単位の差分を抽出し、その差分を更新ファイル107にセーブする。

【0038】続いて、ステップS310で、旧EXEファイル105と新EXEファイル106のそれぞれについて、変数segtbl off及びsegtbl cnt (図6参照) に基づいて抽出されるセグメントテーブル部 (図2参照) が解析される。セグメントテーブル部のデータ構造は、図7に示される。セグメントテーブル部は、セグメントデータの先頭アドレスを示す1ワードのファイルオフセットデータ、セグメントデータの実サイズを示す1ワードの実セグメントデータサイズデータ、及びセグメントデータのタイプ及び属性を示す1ワードのセグメントタイプ/属性データからなるデータセットを、後述するセグメントデータのそれぞれに対応して有している。ステップS310では、旧EXEファイル105と新EXEファイル106のそれぞれについて、このデータ構造が例えばデータ構造体の形式で、セグメント情報108 (図1参照) として、後述するセグメントデータ対応付け部102及びリソース対応付け部103からアクセス可能な特には図示しないメモリに設定される。

【0039】次に、ステップS311で、旧EXEファイル105と新EXEファイル106のそれぞれについて、上述のセグメントテーブル部領域 (図2参照) が抽出される。この領域は、変数segtbl offで示されるセグメントテーブル部の先頭のオフセットアドレスから変数entrytbl offで示されるエクスポートエントリテーブル部の先頭のオフセットアドレスの手前のアドレスまでの領域として抽出される (図6、図2参照)。

【0040】続いて、ヘッダ解析部101は、ステップS312で、差分抽出処理部104をコールすると共に、旧EXEファイル105と新EXEファイル106のそれぞれについて抽出されたセグメントテーブル部領域を差分抽出処理部104に渡す。この結果、差分抽出処理部104は、後述する差分抽出処理により、ヘッダ解析部101から渡されたセグメントテーブル部領域の先頭から末尾まで、新EXEファイル106の旧EXEファイル105に対するバイト単位の差分を抽出し、その差分を更新ファイル107にセーブする。

【0041】次に、ステップS313で、旧EXEファイル105と新EXEファイル106のそれぞれにつき、エクスポートエントリテーブル部領域 (図2参照) が抽出される。この領域は、変数entrytbl offで示されるエクスポートエントリテーブル部の先頭のオフセットアドレスから変数resnamtbl offで示される常駐ネームテーブル部の先頭のオフセットアドレスの手前のアドレ

スまでの領域として抽出される (図6、図2参照)。

【0042】エクスポートエントリテーブル部には、それが含まれるOS/2 EXE形式のプログラムが、他のOS/2プログラムから参照され得るプログラム、例えばDLL (ダイナミックリンクライブラリ) プログラムである場合に、他のOS/2プログラムがこのテーブル部が含まれるプログラムの一部として提供される関数を参照できるように、その関数へのファイル内のセグメント番号とオフセットが格納される。

【0043】続いて、ヘッダ解析部101は、ステップS314で、差分抽出処理部104をコールすると共に、旧EXEファイル105と新EXEファイル106のそれぞれについて抽出されたエクスポートエントリテーブル部領域を差分抽出処理部104に渡す。この結果、差分抽出処理部104は、後述する差分抽出処理により、ヘッダ解析部101から渡されたセグメントテーブル部領域の先頭から末尾まで、新EXEファイル106の旧EXEファイル105に対するバイト単位の差分を抽出し、その差分を更新ファイル107にセーブする。

【0044】次に、ステップS315で、旧EXEファイル105と新EXEファイル106のそれぞれについて、常駐ネームテーブル部及び非常駐ネームテーブル部 (図2参照) の領域が、1つの領域として抽出される。この領域は、変数resnamtbl offで示される常駐ネームテーブル部の先頭のオフセットアドレスから変数modref offで示されるモジュール参照テーブル部の先頭のオフセットアドレスの手前のアドレスまでの領域として抽出される (図6、図2参照)。

【0045】常駐ネームテーブル部及び非常駐ネームテーブル部は、図8に示される共通の構成を有し、常駐ネームテーブル部には、プログラム実行時にメモリに常駐するモジュール名とそのバイト数及びエントリされるテーブルのインデックスが格納され、非常駐ネームテーブル部には、プログラム実行時に必要に応じてそのプログラム自体が呼出すモジュール名とそのバイト数及びエントリされるテーブル (モジュール) のインデックスが格納される。

【0046】続いて、ヘッダ解析部101は、ステップS316で、差分抽出処理部104をコールすると共に、旧EXEファイル105と新EXEファイル106のそれぞれについて抽出された常駐/非常駐ネームテーブル部領域を差分抽出処理部104に渡す。この結果、差分抽出処理部104は、後述する差分抽出処理によって、ヘッダ解析部101から渡された常駐/非常駐ネームテーブル部領域の先頭から末尾まで、新EXEファイル106の旧EXEファイル105に対するバイト単位の差分を抽出し、その差分を更新ファイル107にセーブする。

【0047】次に、ステップS317において、旧EX

10

20

30

40

50

Eファイル105と新EXEファイル106のそれぞれについて、モジュール参照テーブル部及びインポートネームテーブル部（図2参照）の領域が、1つの領域として抽出される。この領域は、変数modref offで示されるモジュール参照テーブル部の先頭のオフセットアドレスから変数resrctbl offで示されるリソーステーブル部の先頭のオフセットアドレスの手前のアドレスまでの領域として抽出される（図6、図2参照）。

【0048】インポートネームテーブル部には、図9に示されるように、それが含まれるプログラムが参照する他のDLLプログラムのモジュール名とそのバイト長が格納され、モジュール参照テーブル部には、図10に示されるように、インポートネームテーブル部の各要素へのオフセット値がワード配列として格納される。

【0049】続いて、ヘッダ解析部101は、ステップS318で、差分抽出処理部104をコールすると共に、旧EXEファイル105と新EXEファイル106のそれぞれについて抽出されたモジュール参照／インポートネームテーブル部領域を差分抽出処理部104に渡す。この結果、差分抽出処理部104は、後述する差分抽出処理によって、ヘッダ解析部101から渡されたモジュール参照／インポートネームテーブル部領域の先頭から末尾まで、新EXEファイル106の旧EXEファイル105に対するバイト単位の差分を抽出し、その差分を更新ファイル107にセーブする。

【0050】次に、ステップS319で、旧EXEファイル105と新EXEファイル106のそれぞれについて、変数resrctbl off及びresrctbl cnt（図6参照）に基づいて抽出されるリソーステーブル部（図2参照）が解析される。リソーステーブル部のデータ構造は、図11に示される。リソーステーブル部は、リソースのタイプの識別子を示す1ワードのリソースタイプIDデータと、リソースネームの識別子を示す1ワードのリソースネームIDデータとからなるデータセットを、後述するリソースのそれぞれに対応して有している。ステップS319では、旧EXEファイル105と新EXEファイル106のそれぞれについて、このデータ構造が、リソース情報109（図1参照）として、後述するリソース対応付け部103からアクセス可能な特には図示しないメモリに設定される。

【0051】次に、ステップS320で、旧EXEファイル105と新EXEファイル106のそれぞれについて、上述のリソーステーブル部領域（図2参照）が抽出される。この領域は、変数resrctbl offで示されるリソーステーブル部の先頭のオフセットアドレスから変数resrctbl cntで示されるリソースエントリ数分だけ進んだアドレスまでの領域として抽出される（図6、図2参照）。

【0052】続いて、ヘッダ解析部101は、ステップS321で、差分抽出処理部104をコールすると共

に、旧EXEファイル105と新EXEファイル106のそれぞれにつき抽出されたリソーステーブル部領域を差分抽出処理部104に渡す。この結果、差分抽出処理部104は、後述する差分抽出処理により、ヘッダ解析部101から渡されたリソーステーブル部領域の先頭から末尾まで、新EXEファイル106の旧EXEファイル105に対するバイト単位の差分を抽出し、その差分を更新ファイル107にセーブする。

【0053】以上のようにして、ヘッダ解析部101によって、DOS EXE形式の領域、OS/2EXEヘッダ領域、及び各テーブル部領域のそれぞれの領域毎に、旧EXEファイル105と新EXEファイル106との差分抽出処理が実行される。

【0054】そして、ヘッダ解析部101は、ステップS322で、図1のセグメントデータ対応付け部102を起動して、処理を終了する。セグメントデータ対応付け部102で実行される処理の動作フローチャートは図12に示される。このセグメントデータ対応付け部102の動作は、ヘッダ解析部101の場合と同様、特に図示しないコンピュータシステムにおいて、CPUが、ディスク装置などに記憶された旧EXEファイル105及び新EXEファイル106に対して、メモリに記憶された制御プログラムを実行する動作の一部として実現される。

【0055】ここで、旧EXEファイル105と新EXEファイル106とで対応するセグメントデータがある場合、変更されていないセグメントについてはそれらの内容が一致し、変更されたセグメントについてののみ差分が生ずる。そこで、セグメントデータ対応付け部102では、旧EXEファイル105と新EXEファイル106との間で、セグメントデータの属性やその主要部のデータを比較することによりその対応付けを行い、その対応付けしたセグメントデータ領域毎に差分抽出処理部104に対して差分抽出処理を指示する。

【0056】まず、ステップS1201で、前述した図4のステップS310でヘッダ解析部101から設定されたセグメント情報108に基づいて、セグメントの対応付けが行われる。

【0057】この処理では、まず、旧EXEファイル105のセグメント情報108（図7参照）に基づいて、旧EXEファイル105のセグメントが一行に並べられる。なお、セグメントは、リンカによってファイルオフセットが増大する方向に並べられている。並べられた旧EXEファイル105のセグメントの例を図13(a)に示す。

【0058】次に、新EXEファイル106のセグメント情報108（図7参照）に基づいて、新EXEファイル106のセグメントが一行に並べられる。並べられた新EXEファイル106のセグメントの例を図13(b)に示す。

10

20

30

40

50

【0059】図13の例では、新EXEファイル106においては、旧EXEファイル105に対して、セグメント番号3のセグメントが追加されている。従って、この状態が検出される必要がある。

【0060】今、本実施例では、旧EXEファイル105から新EXEファイル106への更新は、セグメントを削除、置換、又は追加することにより行われるとする。従って、旧EXEファイル105と新EXEファイル106のセグメントの対応関係が図14(a)に示されるようなものになることはあり得ず、その対応関係は例えば図14(b)に示される如くなる。この例では、旧EXEファイル105のセグメント番号4のセグメントが削除され、新EXEファイル106のセグメント番号2のセグメントが追加され、その他のセグメントのペアについては、変更がないか、又は置換がなされている。

【0061】そこで、一列に並べられた旧EXEファイル105のセグメント及び新EXEファイル106のセグメントについて、以下の対応付け処理がなされる。

処理1：旧EXEファイル105と新EXEファイル106のセグメント数が比較される。

【0062】処理2：セグメント数が同じか、新EXEファイル106のセグメント数の方が多い場合は、旧EXEファイル105で定義されているセグメントをキーとして、新EXEファイル106中のセグメントがサーチされる。旧EXEファイル105のセグメント数の方が多い場合は、新EXEファイル106で定義されているセグメントをキーとして、旧EXEファイル105中のセグメントがサーチされる。

【0063】処理3：サーチ対象であるセグメントにおいて、現在のキーとされているセグメントに対し、セグメントタイプ/属性が同じで、かつ、セグメントデータの最初の16バイトの内容が一致するものがペアセグメントとされる。ここで、図15に示されるように、先頭にリロケーション情報が設定されているリロケーションテーブルを有するセグメントデータが存在する。キーとされているセグメント又はサーチ対象であるセグメントの何れかがこのようなデータである場合には、このリロケーションテーブルの直後の最初の16バイトの内容が一致するものがペアセグメントとされ、リロケーションテーブルについては、後述するように別に差分抽出処理が実行される。なお、リロケーションテーブルが存在するか否かは、セグメント情報108のセグメントタイプ/属性データに、リロケーションテーブルが存在することを示すデータ値が設定されているか否かによって判別できる。

【0064】サーチ対象であるセグメントにおいて、上記一致関係が成立するセグメントがない場合には、キーとなったセグメントについては、サーチは諦められ、そのセグメントは旧EXEファイル105の孤立セグメントとされる。そして、次のセグメントがキーとされて、

サーチが行われる。

【0065】全てのサーチが終了した時点で、新EXEファイル106においてペアとならなかったセグメントは、新EXEファイル106の孤立セグメントとされる。このようにして得られた、ペアセグメント、旧EXEファイル105の孤立セグメント、及び新EXEファイル106の孤立セグメントが、セグメント対応付けデータとしてメモリなどに保持される。

【0066】上述のようにして、図12のステップS1201で、全てのサーチが終了した後、セグメント対応付けデータのそれぞれに対して、以下に示されるステップS1202～S1216の処理が繰り返し実行される。

【0067】即ち、まず、ステップS1202で、1つのセグメント対応付けデータがメモリなどから抽出される。セグメント対応付けデータが存在するならステップS1203の判定はYESとなり、続いて、ステップS1204で、そのセグメント対応付けデータがペアセグメントか否かが判定される。

【0068】セグメント対応付けデータがペアセグメントであってステップS1204の判定がYESの場合には、ステップS1208、S1210、及びS1212で、旧EXEファイル105及び新EXEファイル106の両方又は何れか一方にリロケーション情報が設定されているか否かが判定される。

【0069】旧EXEファイル105及び新EXEファイル106の両方にリロケーション情報が設定されている場合は、ステップS1208及びS1210の判定がNOとなりステップS1212の判定がYESとなつて、ステップS1213が実行される。ステップS1213では、旧EXEファイル105と新EXEファイル106のそれぞれにつき、リロケーションテーブル部領域が抽出される。この領域は、セグメントデータの先頭アドレスから、その先頭1バイトに設定されているリロケーション項目数分だけ進んだアドレスまでの領域として抽出される。続いて、セグメントデータ対応付け部102は、ステップS1214で、差分抽出処理部104をコールすると共に、旧EXEファイル105と新EXEファイル106のそれぞれについて抽出されたリロケーションテーブル部領域を差分抽出処理部104に渡す。この結果、差分抽出処理部104は、後述する差分抽出処理によって、セグメントデータ対応付け部102から渡されたリロケーションテーブル部領域の先頭から末尾まで、新EXEファイル106の旧EXEファイル105に対するバイト単位の差分を抽出し、その差分を更新ファイル107にセーブする。

【0070】旧EXEファイル105のみにリロケーション情報が設定されている場合は、ステップS1208の判定がYESとなり、ステップS1209が実行される。ステップS1209では、旧EXEファイル105

10

20

30

40

50

のリソーステーブル部領域が削除された旨を示す差分情報（更新レコード）が、更新ファイル107にセーブされる。

【0071】新EXEファイル106のみにリロケーション情報が設定されている場合は、ステップS1208の判定がNO、ステップS1210の判定がYESとなり、ステップS1211が実行される。ステップS1211においては、新EXEファイル106のリソーステーブル部領域が挿入された旨を示す差分情報（更新レコード）が、更新ファイル107にセーブされる。

【0072】旧EXEファイル105及び新EXEファイル106の何れにもリロケーション情報が設定されていない場合は、上述のリロケーションテーブルに関する処理は実行されない。

【0073】以上のステップS1214、S1209、又はS1211の処理の後、又はステップS1212の判定がNOとなった後、ステップS1215が実行される。ステップS1215では、リロケーション情報を除いたセグメントデータ本体の領域が抽出される。この領域は、リロケーション情報が設定されていない場合にはセグメントデータの先頭アドレスから、リロケーション情報が設定されている場合にはセグメントデータの先頭1バイトに設定されているリロケーション項目数分だけ進んだアドレスの次のアドレスから、セグメント情報108から抽出される次のセグメント番号のセグメントデータの先頭のオフセットアドレスの手前のアドレスまでの領域として抽出される。

【0074】続いて、セグメントデータ対応付け部102は、ステップS1216で、差分抽出処理部104をコールすると共に、旧EXEファイル105と新EXEファイル106のそれぞれにつき抽出されたセグメントデータ領域を差分抽出処理部104に渡す。この結果、差分抽出処理部104は、後述する差分抽出処理によって、セグメントデータ対応付け部102から渡されたセグメントデータ領域の先頭から末尾まで、新EXEファイル106の旧EXEファイル105に対するバイト単位の差分を抽出し、その差分を更新ファイル107にセーブする。

【0075】一方、前述したステップS1204の判定において、セグメント対応付けデータがペアセグメントでない場合であってステップS1204の判定がNOの場合には、更にステップS1205で、セグメント対応付けデータが旧EXEファイル105の孤立セグメントであるか否かが判定される。

【0076】セグメント対応付けデータが旧EXEファイル105の孤立セグメントであってステップS1205の判定がYESなら、ステップS1206が実行される。ステップS1206では、旧EXEファイル105の孤立セグメントの領域が削除された旨を示す差分情報（更新レコード）が、更新ファイル107にセーブされ

る。

【0077】セグメント対応付けデータが旧EXEファイル105の孤立セグメントでなくステップS1205の判定がNOなら、それは新EXEファイル106の孤立セグメントであり、この場合には、ステップS1207が実行される。ステップS1207では、新EXEファイル106の孤立セグメントの領域が挿入された旨を示す差分情報（更新レコード）が、更新ファイル107にセーブされる。

10 【0078】以上のステップS1202～S1216の処理が、全てのセグメント対応付けデータに対して繰り返し実行される。そして、セグメント対応付けデータがなくなると、ステップS1203の判定がNOとなる。このようにして、セグメントデータ対応付け部102によって、旧EXEファイル105と新EXEファイル106との間で、セグメントデータの対応付けが行われた上で、その対応付けされたセグメントデータ領域毎に差分抽出処理部104で差分抽出処理が実行される。

20 【0079】そして、セグメントデータ対応付け部102は、ステップS1217で、図1のリソース対応付け部103を起動して、処理を終了する。リソース対応付け部103で実行される処理の動作フローチャートは、図16に示される。このリソース対応付け部103の動作は、ヘッダ解析部101などの場合と同様、特に図示しないコンピュータシステムにおいて、CPUが、ディスク装置などに記憶された旧EXEファイル105及び新EXEファイル106に対して、メモリに記憶された制御プログラムを実行する動作の一部として実現される。

30 【0080】ここで、リソースはセグメントデータの属性として規定され、従って、前述したセグメントデータの場合と同様、旧EXEファイル105と新EXEファイル106とで対応するリソースがある場合、変更されていないリソースについてはそれらの内容が一致し、変更されたリソースについてのみ差分が生ずる。そこで、リソース対応付け部103でも、セグメントデータ対応付け部102の場合と同様に、旧EXEファイル105と新EXEファイル106との間で、リソースの対応付けを行い、その対応付けしたリソース領域毎に差分抽出処理部104に対して差分抽出処理を指示する。

40 【0081】この場合、特にリソースでは、前述したリソーステーブル部（図11参照）において、リソースネームIDが設定されている。そこで、リソース対応付け部103は、リソースネームIDを比較することによりその対応付けを行い、その対応付けしたリソース領域毎に差分抽出処理部104に対して差分抽出処理を指示する。

【0082】まず、ステップS1601で、前述した図4のステップS310及びS319でヘッダ解析部101から設定されたセグメント情報108及びリソース情

報109に基づいて、リソースの対応付けが行われる。

【0083】この処理では、まず、旧EXEファイル105と新EXEファイル106のセグメント情報108（図7参照）及びリソース情報109に基づいて、それらのリソースが抽出される。ここで、{(セグメント情報108として設定されているセグメントの数) - (リソース情報109として設定されているリソースの数) + 1} 番目以降のセグメント情報108が、リソース情報109として設定されている各リソースネームIDに対応するリソースを示している。旧EXEファイル105及び新EXEファイル106において抽出されたリソースの例を図17(a)に示す。

【0084】次に、旧EXEファイル105及び新EXEファイル106において抽出されたリソースが、セグメント情報108から抽出されるファイルオフセット順にソートされる。ソートされた旧EXEファイル105及び新EXEファイル106のリソースの例を図17(b)に示す。

【0085】次に、ソートされた旧EXEファイル105及び新EXEファイル106のリソースにおいて、同じリソースネームIDを有するもの同士が対応づけられる。この結果、図17の例における対応関係は、図18に示される如くとなる。この例では、旧EXEファイル105のリソースネームID=6のリソースが削除され、新EXEファイル106のリソースネームID=9及び38のリソースが追加され、その他のリソースのペアについては、変更がないか、又は置換がなされている。

【0086】そこで、ソートされた旧EXEファイル105のリソース及び新EXEファイル106のリソースについて、以下の対応付け処理がなされる。

処理1：旧EXEファイル105と新EXEファイル106のリソース数が比較される。

【0087】処理2：リソース数が同じか、新EXEファイル106のリソース数の多い場合は、旧EXEファイル105で定義されているリソースをキーとして、新EXEファイル106中のリソースがサーチされる。旧EXEファイル105のリソース数の多い場合は、新EXEファイル106で定義されているリソースをキーとして、旧EXEファイル105中のリソースがサーチされる。

【0088】処理3：サーチ対象であるリソースにおいて、現在のキーとされているリソースに対し、リソースネームIDが同じものがペアリソースとされる。ここで、前述したセグメントデータの場合と同様、図15に示されるように、先頭にリロケーション情報が設定されているリロケーションテーブルを有するリソースが存在する。キーとされているリソース又はサーチ対象であるリソースの何れかがこのようなデータである場合には、そのリロケーションテーブルについては、後述するよう

に別に差分抽出処理が実行される。なお、リロケーションテーブルが存在するか否かは、リソースに対応するセグメント情報108のセグメントタイプ/属性データに、リロケーションテーブルが存在することを示すデータ値が設定されているか否かによって判別できる。

【0089】サーチ対象であるリソースにおいて、上記一致関係が成立するリソースがない場合には、キーとなったリソースについては、サーチは諦められ、そのリソースは旧EXEファイル105の孤立リソースとされる。そして、次のリソースがキーとされて、サーチが行われる。

【0090】全てのサーチが終了した時点で、新EXEファイル106においてペアとならなかったリソースは、新EXEファイル106の孤立リソースとされる。このようにして得られた、ペアリソース、旧EXEファイル105の孤立リソース、及び新EXEファイル106の孤立リソースが、リソース対応付けデータとしてメモリなどに保持される。

【0091】上述のようにして、図16のステップS1601で、全てのサーチが終了した後、リソース対応付けデータのそれぞれに対して、以下に示されるステップS1602～S1616の処理が繰り返し実行される。これらの処理波、セグメント対応付けデータに対する図12のステップS1202～S1216の処理と同じである。

【0092】即ち、まず、ステップS1602で、1つのリソース対応付けデータがメモリなどから抽出される。リソース対応付けデータが存在するならばステップS1603の判定はYESとなり、続いて、ステップS1604で、そのリソース対応付けデータがペアリソースか否かが判定される。

【0093】リソース対応付けデータがペアリソースであってステップS1604の判定がYESの場合には、ステップS1608、S1610、及びS1612で、旧EXEファイル105及び新EXEファイル106の両方又は何れか一方にリロケーション情報が設定されているか否かが判定される。

【0094】旧EXEファイル105及び新EXEファイル106の両方にリロケーション情報が設定されている場合は、ステップS1608及びS1610の判定がNOとなりステップS1612の判定がYESとなつて、ステップS1613が実行される。ステップS1613では、旧EXEファイル105と新EXEファイル106のそれぞれにつき、リロケーションテーブル領域が抽出される。この領域は、リソースの先頭アドレスから、その先頭1バイトに設定されているリロケーション項目数分だけ進んだアドレスまでの領域として抽出される。続いて、リソース対応付け部103は、ステップS1614で、差分抽出処理部104をコールすると共に、旧EXEファイル105と新EXEファイル106

のそれぞれについて抽出されたリロケーションテーブル部領域を差分抽出処理部104に渡す。この結果、差分抽出処理部104は、後述する差分抽出処理によって、リソース対応付け部103から渡されたリロケーションテーブル部領域の先頭から末尾まで、新EXEファイル106の旧EXEファイル105に対するバイト単位の差分を抽出し、その差分を更新ファイル107にセーブする。

【0095】旧EXEファイル105のみにリロケーション情報が設定されている場合は、ステップS1608の判定がYESとなり、ステップS1609が実行される。ステップS1609では、旧EXEファイル105のリソーステーブル部領域が削除された旨を示す差分情報（更新レコード）が、更新ファイル107にセーブされる。

【0096】新EXEファイル106のみにリロケーション情報が設定されている場合は、ステップS1608の判定がNO、ステップS1610の判定がYESとなり、ステップS1611が実行される。ステップS1611においては、新EXEファイル106のリソーステーブル部領域が挿入された旨を示す差分情報（更新レコード）が、更新ファイル107にセーブされる。

【0097】旧EXEファイル105及び新EXEファイル106の何れにもリロケーション情報が設定されていない場合は、上述のリロケーションテーブルに関する処理は実行されない。

【0098】以上のステップS1614、S1609、又はS1611の処理の後、又はステップS1612の判定がNOとなった後、ステップS1615が実行される。ステップS1615では、リロケーション情報を除いたリソース本体の領域が抽出される。この領域は、リロケーション情報が設定されていない場合にはリソースの先頭アドレスから、リロケーション情報が設定されている場合にはリソースの先頭1バイトに設定されているリロケーション項目数分だけ進んだアドレスの次のアドレスから、セグメント情報108及びリソース情報108から抽出される次のリソース番号のリソースの先頭のオフセットアドレスの手前のアドレスまでの領域として抽出される。

【0099】続いて、リソース対応付け部103は、ステップS1616で、差分抽出処理部104をコールすると共に、旧EXEファイル105と新EXEファイル106のそれぞれにつき抽出されたリソース領域を差分抽出処理部104に渡す。この結果、差分抽出処理部104は、後述する差分抽出処理によって、リソース対応付け部103から渡されたリソース領域の先頭から末尾まで、新EXEファイル106の旧EXEファイル105に対するバイト単位の差分を抽出し、その差分を更新ファイル107にセーブする。

【0100】一方、前述したステップS1604の判定

において、リソース対応付けデータがペアリソースでない場合であってステップS1604の判定がNOの場合には、更にステップS1605で、リソース対応付けデータが旧EXEファイル105の孤立リソースであるか否かが判定される。

【0101】リソース対応付けデータが旧EXEファイル105の孤立リソースであってステップS1605の判定がYESなら、ステップS1606が実行される。ステップS1606では、旧EXEファイル105の孤立リソースの領域が削除された旨を示す差分情報（更新レコード）が、更新ファイル107にセーブされる。

【0102】リソース対応付けデータが旧EXEファイル105の孤立リソースでなくステップS1605の判定がNOなら、それは新EXEファイル106の孤立リソースであり、この場合には、ステップS1607が実行される。ステップS1607では、新EXEファイル106の孤立リソースの領域が挿入された旨を示す差分情報（更新レコード）が、更新ファイル107にセーブされる。

【0103】以上のステップS1602～S1616の処理が、全てのリソース対応付けデータに対して繰り返し実行される。そして、リソース対応付けデータがなくなると、ステップS1603の判定がNOとなり、全ての処理を終了する。

【0104】最後に、上述したようにしてヘッダ解析部101、セグメントデータ対応付け部102、又はリソース対応付け部103からコールされる差分抽出処理部104の処理について説明する。

【0105】差分抽出処理部104で実行される差分抽出処理の動作フローチャートは図19に、また、差分抽出処理において実行される比較処理の動作フローチャートは図20に示される。これらの動作は、ヘッダ解析部101などの場合と同様、特に図示しないコンピュータシステムにおいて、CPUが、ディスク装置などに記憶された旧EXEファイル105及び新EXEファイル106に対して、メモリに記憶された制御プログラムを実行する動作の一部として実現される。

【0106】今、例えば図22(a)に示されるように、旧EXEファイル105の領域として、バイトデータ列a b . . . c d e . . . からなる領域が指定され、一方、新EXEファイル106の領域として、バイトデータcとdの間にバイトデータ列I I . . . Iが挿入された領域が指定されたとする。

【0107】図19において、まず、ステップS1901で、指定された領域において比較すべきデータがなくなったか否かが判定される。比較すべきデータがある場合、ステップS1902で、新EXEファイル106において指定された領域を比較側、旧EXEファイル105において指定された領域を比較対象側として、図21に示されるように、比較側のバイトデータをキーとして

比較処理が実行される。

【0108】比較処理の動作フローチャートは図20に示される。図20のステップS2001で、特に図示しないメモリに確保されている変数srcendに、比較側の領域における終了位置又は後述する現在の比較側の比較開始位置より1024バイト前方の位置のうち、現在の比較側の比較開始位置から近い方の位置のアドレスが格納される。同様に、特に図示しないメモリに確保されている変数distendに、比較対象側の領域における終了位置又は後述する現在の比較対象側の比較開始位置より1024バイト前方の位置のうち、現在の比較対象側の比較開始位置から近い方の位置のアドレスが格納される。

【0109】次に、ステップS2002で、特に図示しないメモリに確保されている変数変数srcに、現在の比較側の比較開始位置のアドレスが格納される。図22(a)の例では、変数srcには最初は新EXEファイル106の指定領域における先頭のバイトデータaのアドレスが格納される。同様に、特に図示しないメモリに確保される変数distとnextdistに、現在の比較対象側の比較開始位置のアドレスが格納される。図22(a)の例では、変数distとnextdistには最初は旧EXEファイル105の指定領域における先頭のバイトデータaのアドレスが格納される。

【0110】その後、図22(a)に示されるように、新EXEファイル106と旧EXEファイル105とで内容が一致しているバイトデータa b . . . cの範囲では、ステップS2006で変数src、dist、及びnextdistの値が順次インクリメントされながら、ステップS2003→S2004→S2005→S2006→S2003の処理が繰り返されることにより、図22(a)のT1～T3に示されるように、変数src及びdistで示されるアドレスにおける値が一致するバイトデータ同士が順次対応付けられてゆく。

【0111】そして、図22(a)に示されるように、ステップS2006で、比較側である新EXEファイル106において変数srcの値が挿入されたバイトデータI I . . . Iのうち最初のデータのアドレスにされ、比較対象側である旧EXEファイル105において変数distの値がバイトデータdのアドレスにされた後、次にステップS2004が実行される時点で、その判定がNOとなる。

【0112】この後は、ステップS2008で変数distの値がインクリメントされながら、ステップS2004→S2007→S2008→S2009→S2004→S2007→S2008の処理が繰り返されることにより、変数srcで示されるアドレスにおける新EXEファイル106で挿入された第1番目のバイトデータIをキーとして、その値と、順次インクリメントされる変数distで示されるアドレスにおける旧EXEファイル105

上のバイトデータd e . . . の各値とが、それらが一致してステップS2004の判定がYESとなるまで、順次比較される。この動作は、図22(a)のT4、T5に示される。

【0113】この比較処理の繰り返しの結果、新EXEファイル106で挿入された第1番目のバイトデータIに一致するバイトデータが旧EXEファイル105上で見つけれなければ、やがて変数distで示されるアドレスが変数distendで示される比較終了アドレスを越え、ステップS2009の判定がYESとなる。

【0114】この場合には、新EXEファイル106で挿入された第1番目のバイトデータIをキーとするサーチは諦められ、ステップS2010で、変数srcの値がインクリメントされることにより新EXEファイル106で挿入された第2番目のバイトデータIがキーとされ、また、変数distに変数nextdistのアドレスが挿入されることにより、比較対象側である旧EXEファイル105上の比較開始位置が初めて不一致が発生したバイトデータdのアドレスに戻される。

【0115】そして、変数srcで示されるアドレスが変数srcendで示される比較終了アドレスを越えておらずステップS2003の判定がNOであれば、再び、ステップS2008で変数distの値がインクリメントされながら、ステップS2004→S2007→S2008→S2009→S2004→S2007→S2008の処理が繰り返されることにより、変数srcで示されるアドレスにおける新EXEファイル106で挿入された第2番目のバイトデータIをキーとして、その値と、順次インクリメントされる変数distで示されるアドレスにおける旧EXEファイル105上のバイトデータd e . . . の各値とが、それらが一致してステップS2004の判定がYESとなるまで、順次比較される。この動作は、図22(a)のT6、T7に示される。

【0116】以下、ステップS2009の判定がYESとなる毎に、新EXEファイル106で挿入された第3番目以降の各バイトデータIについて同様の比較処理が繰り返される(図22(a)のT8、T9)。

【0117】やがてステップS2010で、変数srcに新EXEファイル106上のバイトデータdのアドレスが代入され、変数distに変数nextdistで示される旧EXEファイル105上のバイトデータdのアドレスが代入された後、ステップS2004が実行された時点で、変数srcで示されるアドレスにおける新EXEファイル106のバイトデータdと変数distで示されるアドレスにおける旧EXEファイル105のバイトデータdとが一致し、その判定がYESとなる。この一致状態は、最初に新EXEファイル106と旧EXEファイル105のバイトデータa b . . . cが一致し、次に、新EXEファイル106で挿入されたバイトデータI I . . . Iが旧EXEファイル105上のバイトデータd e . . . と

不一致となった後の一致状態であるため、ステップS 2005の判定がYESとなる。

【0118】この結果、ステップS 2011で、図22(a)の新EXEファイル106と旧EXEファイル105のバイトデータa b . . . cで示される一致部分の一致開始点(バイトデータaのアドレス)と一致バイト数が算出され、また、新EXEファイル106で挿入されたバイトデータI I . . . Iで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

【0119】上述のように、一致部分と不一致部分のペアが見つかってステップS 2005の判定がYESとなる時点で、また、図26の説明で後述するように、不一致部分と一致部分のペアが見つかってステップS 2007の判定がYESとなる時点で、一致部分の一致開始点と一致バイト数、及び不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了される。

【0120】以上の動作の結果、新EXEファイル106の指定領域の各バイトデータをキーとする旧EXEファイル105の指定領域の各バイトデータに対する図19のステップS 1902の比較処理によって、一致部分と不一致部分のペアが見つかる、次にステップS 1903で、上述の一致バイト数が64バイト以上であるか否かが判定される。

【0121】一致バイト数が64バイト以上であってステップS 1903の判定がYESとなった場合には、ステップS 1909で、上述の不一致部分に関する差分情報が図1の更新ファイル107へライトされる。例えば図22(a)の新EXEファイル106と旧EXEファイル105のバイトデータa b . . . cで示される一致部分の一致バイト数が64バイト以上である場合には、新EXEファイル106において、旧EXEファイル105のバイトデータcで示されるアドレスの直後にバイトデータI I . . . Iが挿入されたことを示す差分情報が更新ファイル107へライトされる。

【0122】一致バイト数が64バイトより少なくステップS 1903の判定がNOとなった場合には、ステップS 1904で、前述したステップS 1902の場合とは逆に、旧EXEファイル105の指定領域を比較側、新EXEファイル106の指定領域を比較対象側とし、比較側のバイトデータをキーとして、前述した図20で示される比較処理が実行される。

【0123】この比較処理の例として、前述した図22(a)に対応する動作が図22(b)に示される。まず、ステップS 2001とステップS 2002で、変数srcendとdistend、変数src、及び変数distとnextdistに、図22(a)の場合と同様の値が設定される。

【0124】その後、前述したステップS 2003→S 2004→S 2005→S 2006→S 2003の処理が繰り返されることにより、図22(b)のT1～T3に

示されるように、変数src及びdistで示されるアドレスにおける値が一致するバイトデータ同士が順次対応付けられてゆく。

【0125】そして、図22(b)に示されるように、ステップS 2006で、比較側である旧EXEファイル105において変数srcの値がdのアドレスにされ、比較対象側である新EXEファイル106において変数distの値が挿入されたバイトデータI I . . . Iのうち最初のデータのアドレスにされた後、次にステップS 2004が実行される時点で、その判定がNOとなる。

【0126】この後は、ステップS 2008で変数distの値がインクリメントされながら、ステップS 2004→S 2007→S 2008→S 2009→S 2004→S 2007→S 2008の処理が繰り返されることによって、変数srcで示されるアドレスにおける旧EXEファイル105のバイトデータdをキーとして、その値と、順次インクリメントされる変数distで示されるアドレスにおける新EXEファイル106上のバイトデータI I . . . Iの各値とが、それらが一致してステップS 2004の判定がYESとなるまで、順次比較される。この動作は、図22(b)のT4～T6に示される。

【0127】この比較処理の繰り返しの結果、ステップS 2008で、変数distに新EXEファイル106上のバイトデータdのアドレスが代入された後、ステップS 2004が実行された時点で、変数srcで示されるアドレスにおける旧EXEファイル105のバイトデータdと変数distで示されるアドレスにおける新EXEファイル106のバイトデータdとが一致し、その判定がYESとなる。この一致状態は、最初に旧EXEファイル105と新EXEファイル106のバイトデータa b . . . cが一致し、次に、新EXEファイル106で挿入されたバイトデータI I . . . Iが旧EXEファイル105上のバイトデータdと不一致となった後の一致状態であるため、ステップS 2005の判定がYESとなる。

【0128】この結果、ステップS 2011で、図22(b)の旧EXEファイル105と新EXEファイル106のバイトデータa b . . . cで示される一致部分の一致開始点(バイトデータaのアドレス)と一致バイト数が算出され、また、新EXEファイル106で挿入されたバイトデータI I . . . Iで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

【0129】以上の動作の結果、旧EXEファイル105の指定領域の各バイトデータをキーとする新EXEファイル106の指定領域の各バイトデータに対する図19のステップS 1904の比較処理によって、一致部分と不一致部分のペアが見つかる、次にステップS 1905で、全てのバイトが不一致であるか否かが判定される。

【0130】全てのバイトが不一致であってステップS

10

20

30

40

50

1905の判定がYESとなった場合には、ステップS1909で、上述の不一致部分に関する差分情報が図1の更新ファイル107へライトされる。

【0131】一方、図22の例のように、全てのバイトが不一致でなくステップS1905の判定がNOとなった場合には、ステップS1906で、まず、次式によって、新EXEファイル106のバイトデータをキーとする前述したステップS1902の比較処理における一致バイト数の割合RNEWOLD が算出される。

【0132】

【数1】 $RNEWOLD = (\text{新} \rightarrow \text{旧の一致バイト数}) / \{ (\text{新} \rightarrow \text{旧の一致バイト数} + (\text{新} \rightarrow \text{旧の不一致バイト数})) \}$

次に、次式によって、旧EXEファイル105のバイトデータをキーとする前述したステップS1904の比較処理における一致バイト数の割合ROLDNEW が算出される。

【0133】

【数2】 $ROLDNEW = (\text{旧} \rightarrow \text{新の一致バイト数}) / \{ (\text{旧} \rightarrow \text{新の一致バイト数} + (\text{旧} \rightarrow \text{新の不一致バイト数})) \}$

そして、次式によって、新EXEファイル106をキーとした場合の一致バイト数の割合RNEWOLD の方が旧EXEファイル105をキーとした場合の一致バイト数の割合ROLDNEW より小さいか否かが判定される。

【0134】

【数3】 $RNEWOLD / ROLDNEW < 1$?

今、更新ファイル107には、不一致部分の差分情報が格納されるため、一致部分はなるべく多い方がよい。従って、新EXEファイル106をキーとした場合の一致バイト数の割合RNEWOLD の方が旧EXEファイル105をキーとした場合の一致バイト数の割合ROLDNEW より小さくステップS1906の判定がYESの場合には、ステップS1907で、旧EXEファイル105をキーとした場合のステップS1904の比較処理の結果が選択され、ステップS1909で、その比較処理によって得られている不一致部分に関する差分情報が図1の更新ファイル107へライトされる。一方、新EXEファイル106をキーとした場合の一致バイト数の割合RNEWOLD の方が旧EXEファイル105をキーとした場合の一致バイト数の割合ROLDNEW 以上であってステップS1906の判定がNOの場合には、ステップS1908で、新EXEファイル106をキーとした場合のステップS1902の比較処理の結果が選択され、ステップS1909で、その比較処理によって得られている不一致部分に関する差分情報が図1の更新ファイル107へライトされる。

【0135】但し、図22(a)と(b)の例では、 $RNEWOLD = ROLDNEW$ となるため、ステップS1908で、新EXEファイル106をキーとした場合のステップS1902の比較処理の結果が選択され、ステップS1909で、その比較処理によって得られている不一致部分に関

する差分情報として、新EXEファイル106において旧EXEファイル105のバイトデータcで示されるアドレスの直後にバイトデータII...Iが挿入されたことを示す差分情報が、更新ファイル107へライトされる。

【0136】ステップS1909の処理の後には、ステップS1910で、次の比較開始位置が設定され、ステップS1901以降の処理に戻る。図22(a)の例では、次の比較開始位置は、新EXEファイル106では、最後に図20のステップS2004の判定がYESとなったときの変数srcに格納されているバイトデータdのアドレス、旧EXEファイル105でも、同じくそのときの変数distに格納されているバイトデータdのアドレスである。以後、これらの比較開始位置を起点として、前述した場合と全く同様の比較処理が繰り返される。

【0137】新EXEファイル106の指定領域と旧EXEファイル105の指定領域において、比較するバイトデータが存在しなくなったら、ステップS1901の判定がYESとなって、これらの指定領域における差分抽出処理を終了する。

【0138】次に、図23(a)に示されるように、新EXEファイル106の指定領域において、旧EXEファイル105の指定領域のバイトデータcとdの間に存在したバイトデータ列DD...Dの領域が削除された場合を考える。

【0139】この場合、まず、図23(a)に示されるように、図19のステップS1902で、新EXEファイル106において指定された領域を比較側、旧EXEファイル105において指定された領域を比較対象側として、比較側のバイトデータをキーとして比較処理が実行される。

【0140】この比較処理は、前述した図22(b)の場合に対し、比較側と比較対象側が逆になっただけで、同様の処理である。そして、この比較処理の結果、図23(a)の新EXEファイル106と旧EXEファイル105のバイトデータa b...cで示される一致部分の一致開始点と一致バイト数が算出され、また、新EXEファイル106で削除されたバイトデータDD...Dで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

【0141】以上の動作の結果、新EXEファイル106の指定領域の各バイトデータをキーとする旧EXEファイル105の指定領域の各バイトデータに対する図19のステップS1902の比較処理によって、一致部分と不一致部分のペアが見つかったと、次にステップS1903で、上述の一致バイト数が64バイト以上であるか否かが判定される。

【0142】一致バイト数が64バイト以上であってステップS1903の判定がYESとなった場合には、ステップS1909で、上述の不一致部分に関する差分情

10

20

30

40

50

報が図1の更新ファイル107へライトされる。

【0143】一致バイト数が64バイトより少なくステップS1903の判定がNOとなった場合には、ステップS1904で、前述したステップS1902の場合とは逆に、旧EXEファイル105の指定領域を比較側、新EXEファイル106の指定領域を比較対象側とし、比較側のバイトデータをキーとして、前述の図20で示される比較処理が実行される。この比較処理の例として、前述の図23(a)に対応する動作が図23(b)に示される。この比較処理は、前述の図22(a)の場合に

対して、比較側と比較対象側が逆になっただけで、同様の処理である。そして、この比較処理の結果、図23(b)の旧EXEファイル105と新EXEファイル106のバイトデータa b . . . cで示される一致部分の一致開始点と一致バイト数が算出され、また、新EXEファイル106で削除されたバイトデータDD . . . Dで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

【0144】以上の動作の結果、旧EXEファイル105の指定領域の各バイトデータをキーとする新EXEファイル106の指定領域の各バイトデータに対する図19のステップS1904の比較処理によって、一致部分と不一致部分のペアが見つかる、次にステップS1905で、全てのバイトが不一致であるか否かが判定される。

【0145】図23の例では、全てのバイトが不一致でなくステップS1905の判定がNOとなるため、ステップS1906で、前述した数1式、数2式、及び数3式に基づいて、新EXEファイル106をキーとした場合の一致バイト数の割合RNEWOLDの方が旧EXEファイル105をキーとした場合の一致バイト数の割合ROLDNEWより小さいか否かが判定される。

【0146】図23(a)と(b)の例では、RNEWOLD=ROLDNEWとなるため、ステップS1908で、新EXEファイル106をキーとした場合のステップS1902の比較処理の結果が選択され、ステップS1909で、その比較処理によって得られている不一致部分に関する差分情報として、新EXEファイル106において旧EXEファイル105のバイトデータcとdの間のバイトデータDD . . . Dが削除されたことを示す差分情報が、更新ファイル107へライトされる。

【0147】次に、図24(a)に示されるように、新EXEファイル106の指定領域において、旧EXEファイル105の指定領域のバイトデータcとdの間に存在したバイトデータ列RR . . . Rの領域がそれと同じ長さを有する他のバイトデータ列RR . . . Rに置換された場合を考える。

【0148】この場合、まず、図24(a)に示されるように、図19のステップS1902で、新EXEファイル106において指定された領域を比較側、旧EXEフ

ァイル105において指定された領域を比較対象側として、比較側のバイトデータをキーとして比較処理が実行される。この比較処理は、まず、図24(a)のT1~T9で示される比較範囲では、前述の図22(a)のT1~T9で示される比較範囲のものと同様の処理であり、続いて、図24(a)のT10~T13で示される比較範囲では、前述の図23(a)のT4~T7で示される比較範囲のものと同様の処理である。そして、この比較処理の結果、図24(a)の新EXEファイル106と旧EXEファイル105のバイトデータa b . . . cで示される一致部分の一致開始点と一致バイト数が算出され、また、新EXEファイル106で置換されたバイトデータRR . . . Rで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

【0149】以上の動作の結果、新EXEファイル106の指定領域の各バイトデータをキーとする旧EXEファイル105の指定領域の各バイトデータに対する図19のステップS1902の比較処理によって、一致部分と不一致部分のペアが見つかる、次にステップS1903で、上述の一致バイト数が64バイト以上であるか否かが判定される。

【0150】一致バイト数が64バイト以上であってステップS1903の判定がYESとなった場合には、ステップS1909で、上述の不一致部分に関する差分情報が図1の更新ファイル107へライトされる。

【0151】一致バイト数が64バイトより少なくステップS1903の判定がNOとなった場合には、ステップS1904で、前述したステップS1902の場合とは逆に、旧EXEファイル105の指定領域を比較側、新EXEファイル106の指定領域を比較対象側とし、比較側のバイトデータをキーとして、前述の図20で示される比較処理が実行される。この比較処理の例として、前述の図24(a)に対応する動作が図24(b)に示される。この比較処理は、まず、図24(b)のT1~T9で示される比較範囲では、前述した図23(b)のT1~T9で示される比較範囲のものと同様の処理であって、続いて、図24(b)のT10~T13で示される比較範囲では、前述した図22(b)のT4~T7で示される比較範囲のものと同様の処理である。そして、この比較処理の結果、図24(b)の旧EXEファイル105と新EXEファイル106のバイトデータa b . . . cで示される一致部分の一致開始点と一致バイト数が算出され、また、新EXEファイル106で置換されたバイトデータRR . . . Rで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

【0152】以上の動作の結果、旧EXEファイル105の指定領域の各バイトデータをキーとする新EXEファイル106の指定領域の各バイトデータに対する図19のステップS1904の比較処理によって、一致部分

と不一致部分のペアが見つかり、次にステップS1905で、全てのバイトが不一致であるか否かが判定される。

【0153】図24の例では、全てのバイトが不一致でなくステップS1905の判定がNOとなるため、ステップS1906で、前述した数1式、数2式、及び数3式に基づいて、新EXEファイル106をキーとした場合の一致バイト数の割合RNEWOLDの方が旧EXEファイル105をキーとした場合の一致バイト数の割合ROLDNEWより小さいか否かが判定される。

【0154】図24(a)と(b)の例では、RNEWOLD=ROLDNEWとなるため、ステップS1908で、新EXEファイル106をキーとした場合のステップS1902の比較処理の結果が選択され、ステップS1909で、その比較処理によって得られている不一致部分に関する差分情報として、新EXEファイル106において旧EXEファイル105のバイトデータcとdの間のバイトデータRR・・・Rが置換されたことを示す差分情報が、更新ファイル107へライトされる。

【0155】次に、図25(a)に示されるように、新EXEファイル106の指定領域において、旧EXEファイル105の指定領域のバイトデータcとdの間に存在するバイトデータ列のうち、バイトデータ列R・・・R及びr・・・rの領域がそれと同じ長さを有する他のバイトデータ列R・・・R及びr・・・rに置換され、バイトデータ列DD・・・Dが削除され、また、旧EXEファイル105の指定領域のバイトデータcとdの間に、そこに存在していなかった新たなバイトデータ列I I・・・Iが挿入された場合を考える。

【0156】この場合、まず、図25(a)に示されるように、図19のステップS1902で、新EXEファイル106において指定された領域を比較側、旧EXEファイル105において指定された領域を比較対象側として、比較側のバイトデータをキーとして比較処理が実行される。この比較処理は、図24(a)の場合と同様であり、従って、図25(a)のT1～T10で示される比較範囲では、前述の図22(a)のT1～T9で示される比較範囲のものと同様の処理であり、続いて、図25(a)のT11～T13で示される比較範囲では、前述の図23(a)のT4～T7で示される比較範囲のものと同様の処理である。そして、この比較処理の結果、図25(a)の新EXEファイル106と旧EXEファイル105のバイトデータab・・・cで示される一致部分の一致開始点と一致バイト数が算出され、また、新EXEファイル106で挿入、削除、及び置換されたバイトデータI・・・IR・・・Rr・・・rで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

【0157】以上の動作の結果、新EXEファイル106の指定領域の各バイトデータをキーとする旧EXEフ

ァイル105の指定領域の各バイトデータに対する図19のステップS1902の比較処理によって、一致部分と不一致部分のペアが見つかり、次にステップS1903で、上述の一致バイト数が64バイト以上であるか否かが判定される。

【0158】一致バイト数が64バイト以上であってステップS1903の判定がYESとなった場合には、ステップS1909で、上述の不一致部分に関する差分情報が図1の更新ファイル107へライトされる。

10 【0159】一致バイト数が64バイトより少なくステップS1903の判定がNOとなった場合には、ステップS1904で、前述したステップS1902の場合とは逆に、旧EXEファイル105の指定領域を比較側、新EXEファイル106の指定領域を比較対象側とし、比較側のバイトデータをキーとして、前述の図20で示される比較処理が実行される。この比較処理の例として、前述の図25(a)に対応する動作が図25(b)に示される。この比較処理は、前述の図24(b)の場合と同様であり、まず、図25(b)のT1～T9で示される比較範囲では、前述した図23(b)のT1～T9で示される比較範囲のものと同様の処理であって、続いて、図25(b)のT10～T12で示される比較範囲では、前述した図22(b)のT4～T7で示される比較範囲のものと同様の処理である。そして、この比較処理の結果、図25(b)の旧EXEファイル105と新EXEファイル106のバイトデータab・・・cで示される一致部分の一致開始点と一致バイト数が算出され、また、新EXEファイル106で挿入、削除、及び置換されたバイトデータI・・・IR・・・Rr・・・rで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

30 【0160】以上の動作の結果、旧EXEファイル105の指定領域の各バイトデータをキーとする新EXEファイル106の指定領域の各バイトデータに対する図19のステップS1904の比較処理によって、一致部分と不一致部分のペアが見つかり、次にステップS1905で、全てのバイトが不一致であるか否かが判定される。

40 【0161】図25の例では、全てのバイトが不一致でなくステップS1905の判定がNOとなるため、ステップS1906で、前述した数1式、数2式、及び数3式に基づいて、新EXEファイル106をキーとした場合の一致バイト数の割合RNEWOLDの方が旧EXEファイル105をキーとした場合の一致バイト数の割合ROLDNEWより小さいか否かが判定される。

50 【0162】図25(a)と(b)の例では、RNEWOLD=ROLDNEWとなるため、ステップS1908で、新EXEファイル106をキーとした場合のステップS1902の比較処理の結果が選択され、ステップS1909で、その比較処理によって得られている不一致部分に関する差分

情報が更新ファイル107へライトされる。

【0163】ここで、図25(a)において、旧EXEファイル105の指定領域におけるバイトデータcとdの間のバイト数の方が、新EXEファイル106の指定領域におけるバイトデータcとdの間のバイト数より多ければ、まず、新EXEファイル106における当該バイト数分のバイトデータが置換された旨の差分情報が更新ファイル107へライトされ、続いて、{(旧EXEファイル105における当該バイト数) - (新EXEファイル106における当該バイト数)}分のバイトデータが削除された旨の差分情報が更新ファイル107へライトされる。

【0164】逆に、図25(a)において、旧EXEファイル105の指定領域におけるバイトデータcとdの間のバイト数の方が、新EXEファイル106の指定領域におけるバイトデータcとdの間のバイト数より少なければ、まず、旧EXEファイル105における当該バイト数分のバイトデータが置換された旨の差分情報が更新ファイル107へライトされ、続いて、{(新EXEファイル106における当該バイト数) - (旧EXEファイル105における当該バイト数)}分のバイトデータが挿入された旨の差分情報が更新ファイル107へライトされる。

【0165】また、図25(a)において、旧EXEファイル105の指定領域におけるバイトデータcとdの間のバイト数が、新EXEファイル106の指定領域におけるバイトデータcとdの間のバイト数と等しければ、旧EXEファイル105又は新EXEファイル106における当該バイト数分のバイトデータが置換された旨の差分情報が更新ファイル107へライトされる。

【0166】最後に、図26に示されるような特殊な例を考える。この例では、新EXEファイル106の指定領域において、旧EXEファイル105の指定領域の先頭のバイトデータaの手前にバイトデータ列II...Iが挿入され、旧EXEファイル105の指定領域のバイトデータcとdの間に存在したバイトデータ列DD...Dが削除された場合であって、挿入されたバイトデータ列II...Iと削除されたバイトデータ列DD...Dが偶然に一致しているような場合が示されている。

【0167】この場合、まず、図26(a)に示されるように、図19のステップS1902で、新EXEファイル106において指定された領域を比較側、旧EXEファイル105において指定された領域を比較対象側として、比較側のバイトデータをキーとして比較処理が実行される。この比較処理は、図26(a)のT1~T3で示される比較範囲では、前述の図23(a)のT4~T6で示される比較範囲のものと同様の処理であり、続いて、図26(a)のT4~T6で示される比較範囲では、前述の図22(a)のT1~T3で示される比較範囲のもの

同様の処理である。なお、図26(a)のT3→T4の变化時には、最初に新EXEファイル106上の挿入された第1番目のバイトデータIと旧EXEファイル105上の各バイトデータab...cとが不一致となった後に、新EXEファイル106上の挿入された第1番目のバイトデータIと旧EXEファイル105上の新EXEファイル106において削除された第1番目のバイトデータDとが一致した状態となるため、ステップS2005の判定はNOとなる。

【0168】そして、図26(a)のT6で示される処理の後、ステップS2006で、変数srcに新EXEファイル106上のバイトデータaのアドレスが代入され、変数distに旧EXEファイル105上のバイトデータdのアドレスが代入されて、ステップS2004が実行された時点で、変数srcで示されるアドレスにおける新EXEファイル106のバイトデータaと変数distで示されるアドレスにおける旧EXEファイル105のバイトデータdとが不一致となり、その判定がNOとなる。この不一致状態は、最初に新EXEファイル106上の挿入された各バイトデータII...Iと旧EXEファイル105上の各バイトデータab...cとが不一致となり、更に、新EXEファイル106上の挿入された各バイトデータII...Iと旧EXEファイル105上の新EXEファイル106において削除された各バイトデータDD...Dとが一致した後の不一致状態であるため、ステップS2007の判定はYESとなって、ステップS2011が実行される。ステップS2011では、図26(a)の旧EXEファイル105上のバイトデータab...cで示される不一致部分の不一致開始点と不一致バイト数が算出され、また、新EXEファイル106上の挿入されたバイトデータII...Iと旧EXEファイル105上の新EXEファイル106で削除されたバイトデータDD...Dとで示される一致部分の一致開始点と一致バイト数が算出されて、比較処理が終了する。

【0169】以上の動作の結果、新EXEファイル106の指定領域の各バイトデータをキーとする旧EXEファイル105の指定領域の各バイトデータに対する図19のステップS1902の比較処理によって、一致部分と不一致部分のペアが見つかり、次にステップS1903で、上述の一致バイト数が64バイト以上であるか否かが判定される。

【0170】今、図26(a)において、一致部分であるバイトデータ列II...I(=DD...D)のバイト数が例えば数バイト程度と少なく、不一致部分であるバイトデータ列ab...cのバイト数が多い場合には、一致部分が短く不一致部分が長い結果となる。

【0171】このような場合には、ステップS1903の判定がNOとなる結果、ステップS1904で、前述したステップS1902の場合とは逆に、旧EXEファ

イル105の指定領域を比較側、新EXEファイル106の指定領域を比較対象側とし、比較側のバイトデータをキーとして、前述の図20で示される比較処理が実行される。この比較処理の例として、前述の図26(a)に対応する動作が図26(b)に示される。この比較処理は、図26(b)のT1~T3で示される比較範囲では、前述の図22(b)のT4~T6で示される比較範囲のものと同様の処理であり、続いて、図26(b)のT4~T6で示される比較範囲では、前述の図22(b)のT1~T3で示される比較範囲のものと同様の処理である。

【0172】そして、図26(b)のT6で示される処理の後、ステップS2006で、変数srcに旧EXEファイル105上のバイトデータDのアドレスが代入され、変数distに新EXEファイル106上のバイトデータdのアドレスが代入されて、ステップS2004が実行された時点で、変数srcで示されるアドレスにおける旧EXEファイル105のバイトデータDと変数distで示されるアドレスにおける新EXEファイル106のバイトデータdとが不一致となり、その判定がNOとなる。この不一致状態は、最初に旧EXEファイル105のバイトデータaと新EXEファイル106上の挿入された各バイトデータII...Iとが不一致となり、更に、旧EXEファイル105上の各バイトデータab...cと新EXEファイル106上の各バイトデータab...cとが一致した後の不一致状態であるため、ステップS2007の判定はYESとなって、ステップS2011が実行される。ステップS2011では、図26(b)の新EXEファイル106上のバイトデータII...Iで示される不一致部分の不一致開始点と不一致バイト数が算出され、また、旧EXEファイル105と新EXEファイル106上のバイトデータab...cで示される一致部分の一致開始点と一致バイト数が算出されて、比較処理が終了する。

【0173】以上の動作の結果、旧EXEファイル105の指定領域の各バイトデータをキーとする新EXEファイル106の指定領域の各バイトデータに対する図19のステップS1904の比較処理によって、一致部分と不一致部分のペアが見つかったと、次にステップS1905で、全てのバイトが不一致であるか否かが判定される。

【0174】図26の例では、全てのバイトが不一致でなくステップS1905の判定がNOとなるため、ステップS1906で、前述した数1式、数2式、及び数3式に基づいて、新EXEファイル106をキーとした場合の一致バイト数の割合RNEWOLDの方が旧EXEファイル105をキーとした場合の一致バイト数の割合ROLDNEWより小さいか否かが判定される。

【0175】今、前述したように、図26(b)において、一致部分であるバイトデータ列ab...cのバイト数が多く、不一致部分であるバイトデータ列II...I

・Iのバイト数が例えば数バイト程度と少ない場合には、一致部分が長く不一致部分が短い結果となる。

【0176】従って、図26(a)と(b)の例では、RNEWOLD < ROLDNEWとなる。前述したように、更新ファイル107には、不一致部分の差分情報が格納されるため、一致部分はなるべく多い方がよい。従って、図26の場合のように、新EXEファイル106をキーとした場合の一致バイト数の割合RNEWOLDの方が旧EXEファイル105をキーとした場合の一致バイト数の割合ROLDNEWより小さくステップS1906の判定がYESの場合には、ステップS1907で、旧EXEファイル105をキーとした場合のステップS1904の比較処理の結果が選択され、ステップS1909で、その比較処理によって得られている不一致部分に関する差分情報として、新EXEファイル106において旧EXEファイル105のバイトデータaで示されるアドレスの手前にバイトデータII...Iが挿入されたことを示す差分情報が、更新ファイル107へライトされる。

【0177】以上説明したようにして、差分抽出処理部104において差分抽出処理が実行される。図27に、更新ファイル107のデータ構造を示す。

【0178】更新ファイル107の各レコードは、図27(a)に示されるように、そのレコードの属性を示すデータRdCORDIDと、旧EXEファイル105に対するそのレコードの処理位置を示すデータDataOffset、更新される実データの長さを示すデータDataLength及び更新される実データ配列Data n とから構成される。

【0179】RdCORDIDの属性は、図27(b)に示される。まず、値が0のRdCORDIDは、そのレコードが置換レコードであることを示し、この場合には、旧EXEファイル105上のデータDataOffsetで示される位置から、データDataLengthで示される長さのデータが、実データ配列Data n で示されるデータに置換される。

【0180】また、値が1のRdCORDIDは、そのレコードが挿入レコードであることを示し、この場合には、旧EXEファイル105上のデータDataOffsetで示される位置から、データDataLengthで示される長さの実データ配列Data n で示されるデータが挿入される。

【0181】更に、値が2のRdCORDIDは、そのレコードが削除レコードであることを示し、この場合には、旧EXEファイル105上のデータDataOffsetで示される位置から、データDataLengthで示される長さのデータが削除される。なお、この場合には、実データ配列Data n は付加されていない。

【0182】このように生成された更新ファイル107に基づいて旧EXEファイル105が更新される場合の動作は、更新ファイル107から1つ1つレコードが取り出され、各レコードに対して上述の各レコード属性毎の処理が実行されることにより、簡単に実現される。

【0183】以上説明した実施例は、本発明をOS/2シス

10

20

30

40

50

テムのもとで実行可能なEXEファイルの差分抽出処理に適用したものであって、DOS EXE形式の領域、OS/2 EXEヘッダ領域、各テーブル部領域、セグメントデータ領域、及びリソース領域のそれぞれの論理単位毎に、対応付けが行われた上で差分抽出処理が実行される。しかし、本発明はこれに限られるものではなく、他の論理単位、例えばページデータ単位などで対応付けが行われた上で差分抽出処理が実行されるように構成することもできる。

【0184】

【発明の効果】本発明によれば、差分抽出処理が実行される場合、例えばOS/2システムにおける実行ファイルでは、DOS EXE形式の領域、OS/2 EXEヘッダ領域、各テーブル部領域、セグメントデータ領域、及びリソース領域のそれぞれの論理単位毎に、対応付けが行われた上で差分抽出処理が実行されるため、旧実行ファイルと新実行ファイルとの間で、各領域のデータが一致する確率が高くなり、その結果、生成される更新ファイルのサイズを大幅に削減することが可能となる。

【0185】これにより、例えばネットワークを介して更新ファイルが配信される場合などで、新EXEファイルがそのまま配信される場合に比較して回線の使用効率を大幅に向上させることが可能となる。

【図面の簡単な説明】

【図1】本発明の実施例の構成図である。

【図2】本発明の実施例におけるファイル形式を示した図である。

【図3】ヘッダ解析部の動作フローチャート（その1）である。

【図4】ヘッダ解析部の動作フローチャート（その2）である。

【図5】DOS EXEヘッダとOS/2 EXEヘッダのデータ構造を示した図である。

【図6】OS/2 EXEヘッダ情報を示した図である。

【図7】セグメントテーブル部のデータ構造を示した図である。

【図8】常駐／非常駐ネームテーブル部のデータ構造を示した図である。

【図9】インポートネームテーブル部のデータ構造を示した図である。

【図10】モジュール参照テーブル部とインポートネー

ムテーブル部の関係を示した図である。

【図11】リソーステーブル部のデータ構造を示した図である。

【図12】セグメントデータ対応付け部の動作フローチャートである。

【図13】旧EXEファイル及び新EXEファイルのセグメントの配列を示した図である。

【図14】セグメントの対応関係の説明図である。

【図15】リロケーション情報の説明図である。

【図16】リソース対応付け部の動作フローチャートである。

【図17】旧EXEファイルと新EXEファイルのリソースの配列を示した図である。

【図18】リソースの対応関係の説明図である。

【図19】差分抽出処理の動作フローチャートである。

【図20】比較処理の動作フローチャートである。

【図21】比較処理の説明図である。

【図22】新EXEファイルに挿入が行われた場合の差分抽出処理の説明図である。

【図23】新EXEファイルで削除が行われた場合の差分抽出処理の説明図である。

【図24】新EXEファイルで置換が行われた場合の差分抽出処理の説明図である。

【図25】新EXEファイルで挿入、削除、及び置換が同時に行われた場合の差分抽出処理の説明図である。

【図26】新EXEファイルで特殊な挿入及び削除が同時に行われた場合の差分抽出処理の説明図である。

【図27】更新ファイルのデータ構造を示した図である。

【図28】ファイル更新の説明図である。

【図29】ファイル更新環境の説明図である。

【符号の説明】

101	ヘッダ解析部
102	セグメントデータ対応付け部
103	リソース対応付け部
104	差分抽出処理部
105	旧EXEファイル
106	新EXEファイル
107	更新ファイル
108	セグメント情報
109	リソース情報

10

20

30

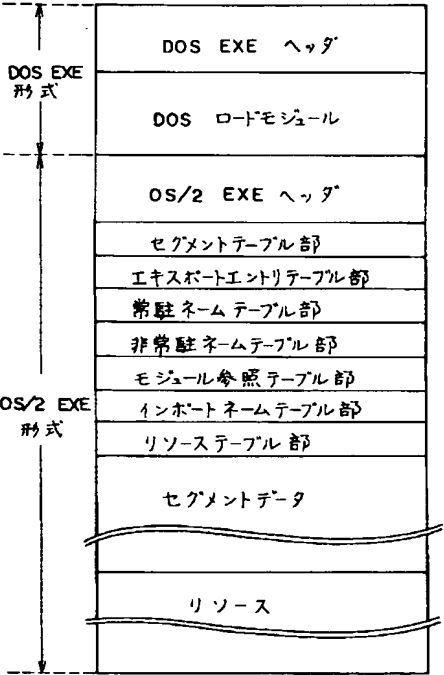
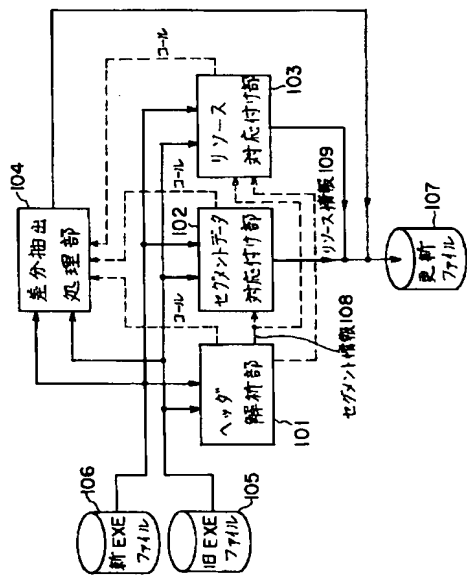
40

【図1】

【図2】

本発明の実施例の構成図

本発明の実施例におけるファイル形式を示した図



【図8】

【図13】

常駐/非常駐ネームテーブル部のデータ構造を示した図

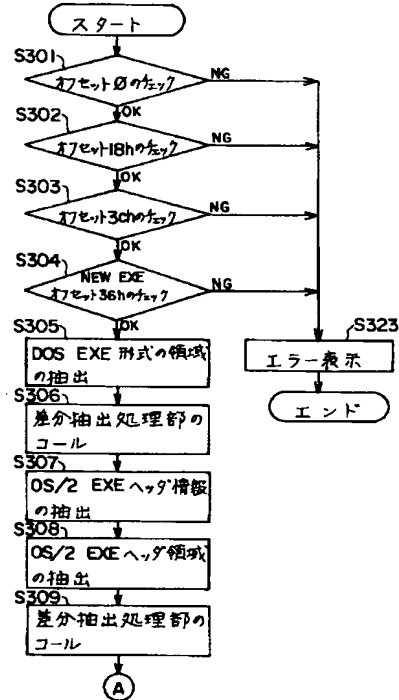
旧EXEファイル及び新EXEファイルのセグメントの配列を示した図

サイズ	オフセット	内容
1バイト	0h	文字列のバイト数、0=テーブルエンド
nバイト	1h	名前列
1ワード	n+1h	エントリテーブルのインデックス

(a) 旧EXEファイル				(b) 新EXEファイル			
Seg No.	offset	Size	Type	Seg No.	offset	Size	Type
1	1123	8000	CODE, LOADUNCALL...	1	1123	8000	CODE, LOADUNCALL...
2	2000	3000	DATA...	2	2000	3000	DATA...
3	3000	4000	CODE, PRELOAD...	3	3000	4000	DATA...
4	3500	3000	DATA...	4	4000	4000	CODE, PRELOAD...

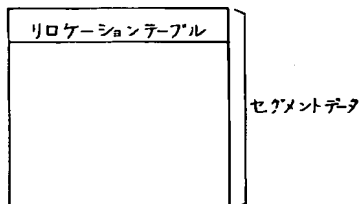
【図3】

ヘッダ解析部の動作フローチャート(その1)



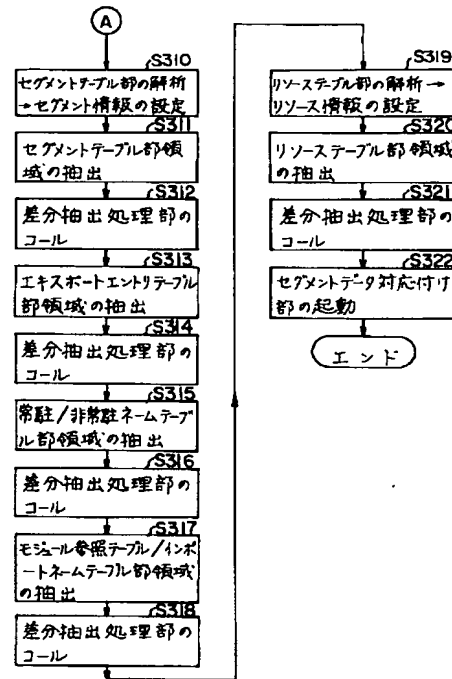
【図15】

リロケーション情報の説明図



【図4】

ヘッダ解析部の動作フローチャート(その2)



【図17】

旧 EXE ファイルと新 EXE ファイルのセグメントの
配列を示した図

(a) ソート前

旧ファイル	
0-4	offset=1000
0-7	offset=3000
0-6	offset=4000
0-5	offset=5000
0-10	offset=6000
0-15	offset=7000
0-30	offset=7000
0-6	offset=2000

新ファイル	
0-4	offset=1000
0-7	offset=3000
0-5	offset=3000
0-10	offset=6000
0-15	offset=7000
0-30	offset=7000
0-8	offset=2000
0-38	offset=6000
0-9	offset=3500

(b) ソート後

旧ファイル	
0-4	offset=1000
0-8	offset=2000
0-7	offset=3000
0-6	offset=4000
0-15	offset=5000
0-10	offset=6000
0-30	offset=7000
0-5	offset=8000

新ファイル	
0-4	offset=1000
0-8	offset=2000
0-7	offset=3000
0-9	offset=3500
0-15	offset=5000
0-10	offset=6000
0-30	offset=7000
0-5	offset=8000

【図5】

DOS EXEヘッダとOS/2 EXEヘッダのデータ構造を示した図

サイズ	オフセット	内 容	DOS EXE ヘッダ の一部
1ワード	0h	識別子 'MZ'	
		:	
1ワード	18h	OS/2 EXE 識別子=0x40	
		:	
1ワード	3Ch	OS/2 EXEへのオフセット	

サイズ	オフセット	内 容	OS/2 EXE ヘッダの 一部
1ワード	0h	識別子 'NE'	
1バイト	2h	リンカのバージョン番号	
1バイト	3h	リンカのリビジョン番号	
● 1ワード	4h	エクスポートエントリテーブルのOS/2 EXE相対オフセット	
● 1ワード	6h	エクスポートエントリテーブルバイトサイズ	
2ワード	8h	CRC-32	
1ワード	C h	フラグ値 8xxxh → DLL 00xxxh → Not Specified 01xxxh → Not Window Compat. (フルスクリーン専用) 02xxxh → Window Compat. (PMセッションで動作可) 03xxxh → Window API (PMのAT)	
1ワード	E h	自動データセグメントのセグメント番号	
1ワード	10h	ローカルヒープサイズ	
1ワード	12h	スタックサイズ	
2ワード	14h	CS:IPアドレス	
2ワード	18h	SS:SPアドレス	
● 1ワード	1Ch	セグメントテーブルの項目数	
1ワード	1Eh	モジュール参照テーブルのエントリ数	
● 1ワード	20h	非常駐ネームテーブルのバイトサイズ	
● 1ワード	22h	セグメントテーブルのOS/2 EXE相対オフセット	
● 1ワード	24h	リソーステーブルのOS/2 EXE相対オフセット	
● 1ワード	26h	常駐ネームテーブルのOS/2 EXE相対オフセット	
● 1ワード	28h	モジュール参照テーブルのOS/2 EXE相対オフセット	
● 1ワード	2Ah	インポートネームテーブルのOS/2 EXE相対オフセット	
● 2ワード	2Ch	非常駐ネームテーブルのファイルの先頭からのオフセット	
1ワード	30h	移動可能エントリの数	
1ワード	32h	セグメント境界シフトカウント (2 ^x のx値)	
● 1ワード	34h	リソーステーブルエントリ数	
1バイト	36h	OSタイプ (1=OS/2)	

●のついている値は、旧ファイルと新ファイルのものをそれぞれ変数内に保持する。
 37h以降は、Windowsのみで使用されているエリアであり、OS/2では使用されない。
 従って、単純比較により差分を抽出する。

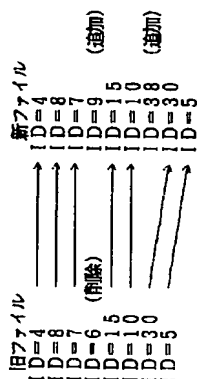
【図6】

OS/2 EXE ヘッド情報も示した図

変数	データの内容
entrytbl.off	エクスポートリテラブルのOS/2 EXE 相対オフセット
entrytbl.len	エクスポートリテラブルのバイトサイズ
segtbl.cnt	セグメントテーブルの項目数
segtbl.off	セグメントテーブルのOS/2 EXE 相対オフセット
resnamtbl.off	常駐ネームテーブルのOS/2 EXE 相対オフセット
modrftbl.off	モジュール参照テーブルのOS/2 EXE 相対オフセット
importtbl.off	インポートネームテーブルのOS/2 EXE 相対オフセット
nonresnam.len	非常駐ネームテーブルの長さ
nonresnam.off	非常駐ネームテーブルのファイル先頭からのオフセット
resrcttbl.off	リソーステーブルのOS/2 EXE 相対オフセット
resrcttbl.cnt	リソーステーブルのエントリ数

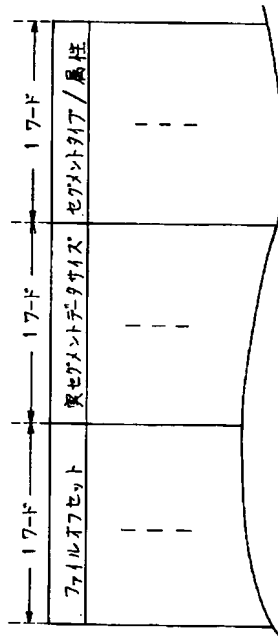
【図18】

リソースの対応関係の説明図



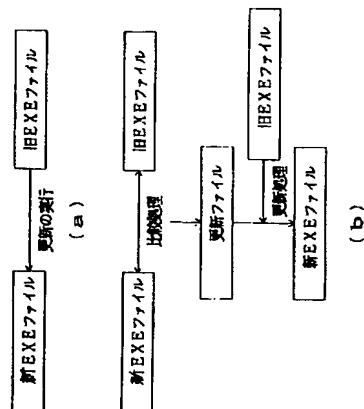
【図7】

セグメントテーブル部のデータ構造を示した図



【図28】

ファイル更新の説明図



【図9】

インポートネームテーブル部のデータ構造も示した図

サイズ	オフセット	内容
1バイト	0h	文字列のバイト長、0-タープアルエンド
nバイト	1h	名前列

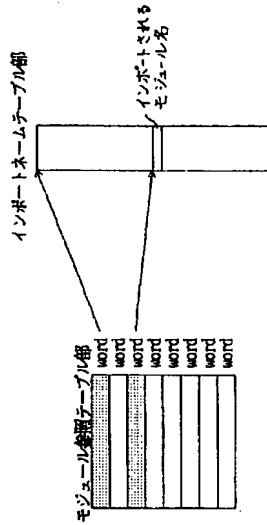
【図11】

リソーステーブル部のデータ構造を示した図

サイズ	オフセット	内容
1ワード	0h	リソースのタイプID
1ワード	2h	リソースのネームID

【図10】

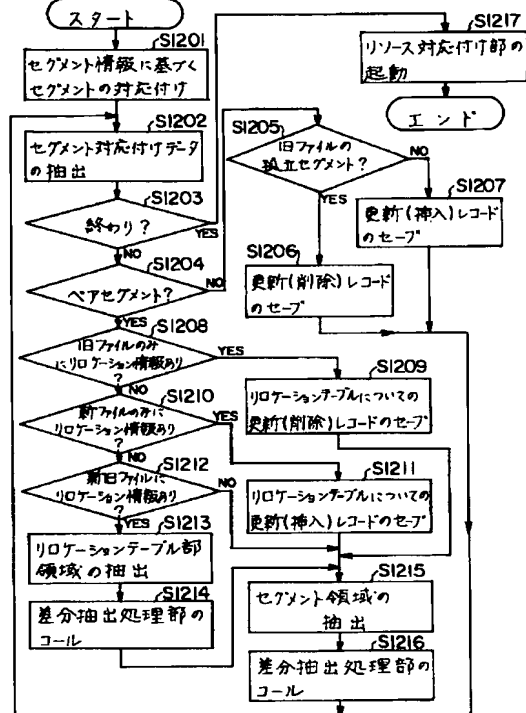
モジュール参照テーブル部とインポートネームテーブル部の関係を示した図



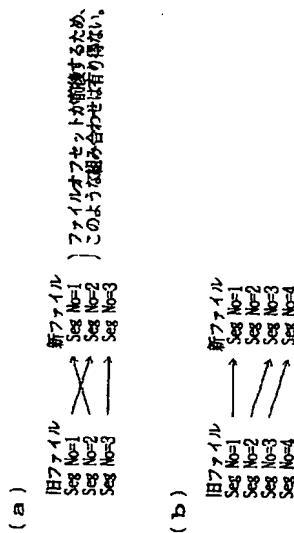
【図14】

【図12】

セグメントデータ対応付け部の動作フローチャート



セグメントの対応関係の説明図



【図27】

更新ファイルのデータ構造を示した図

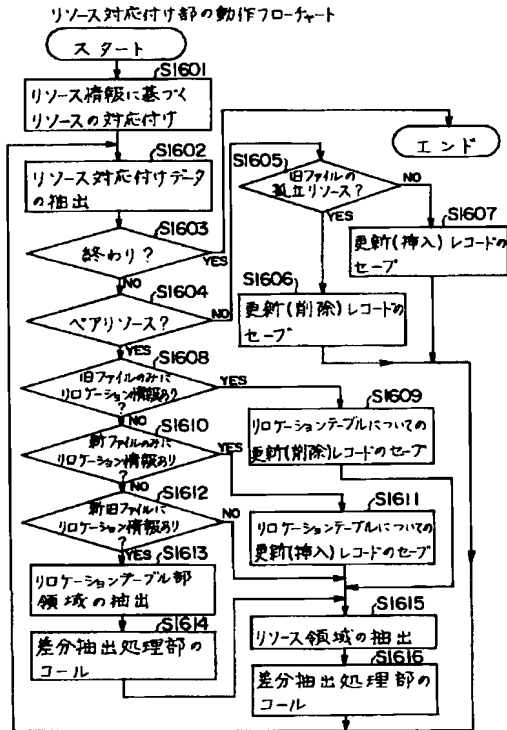
(a) ファイル構造

Record ID	Data Offset	Data Length	Data [n]
0
1
2

(b) RecordIDの意味

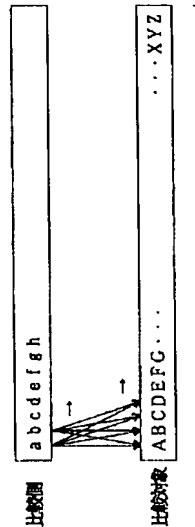
Record ID	意味	備考
0	置換レコード	DataLengthは、置換/挿入するデータ長を表す。
1	挿入レコード	この場合、DataLength 何バイト削除するのかわかり、Dataは存在しない。
2	削除レコード	

【図16】



【図21】

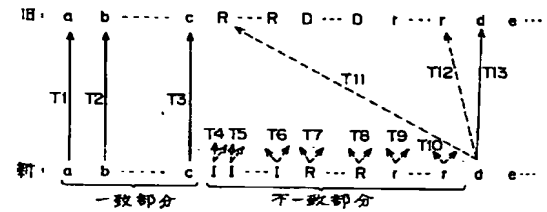
比較処理の説明図



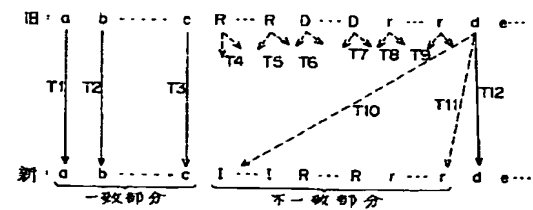
【図25】

新EXEファイル挿入、削除、及び置換が同時に行われた
場合の差分抽出処理の説明図

(a) 新EXEファイル → 旧EXEファイル

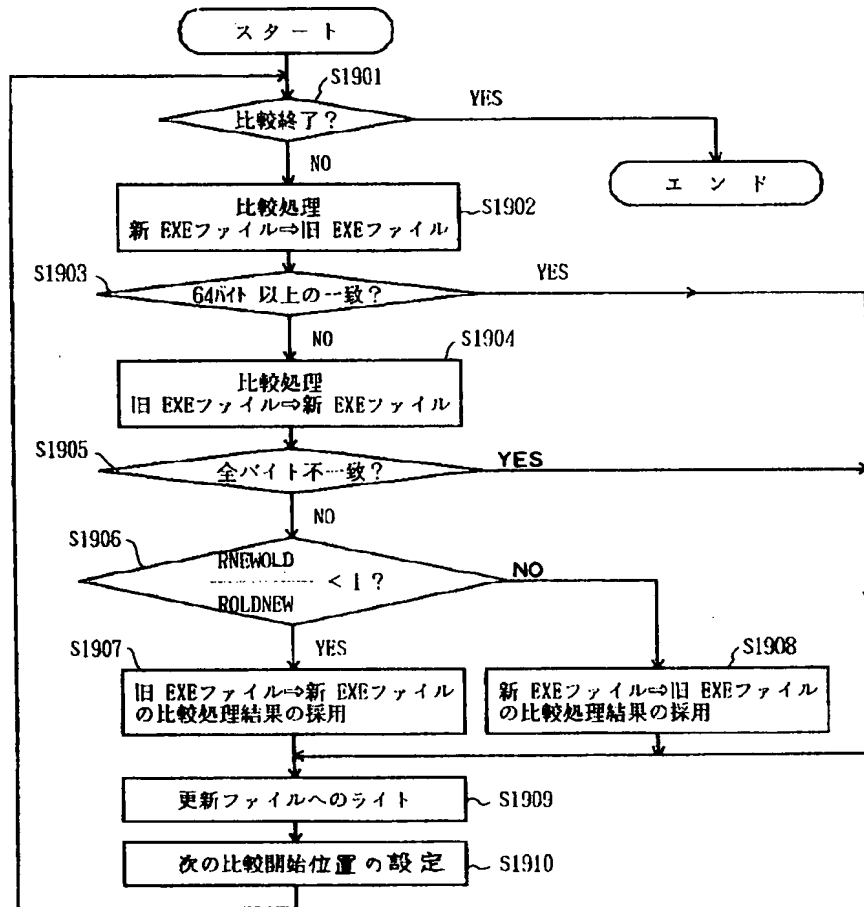


(b) 旧EXEファイル → 新EXEファイル



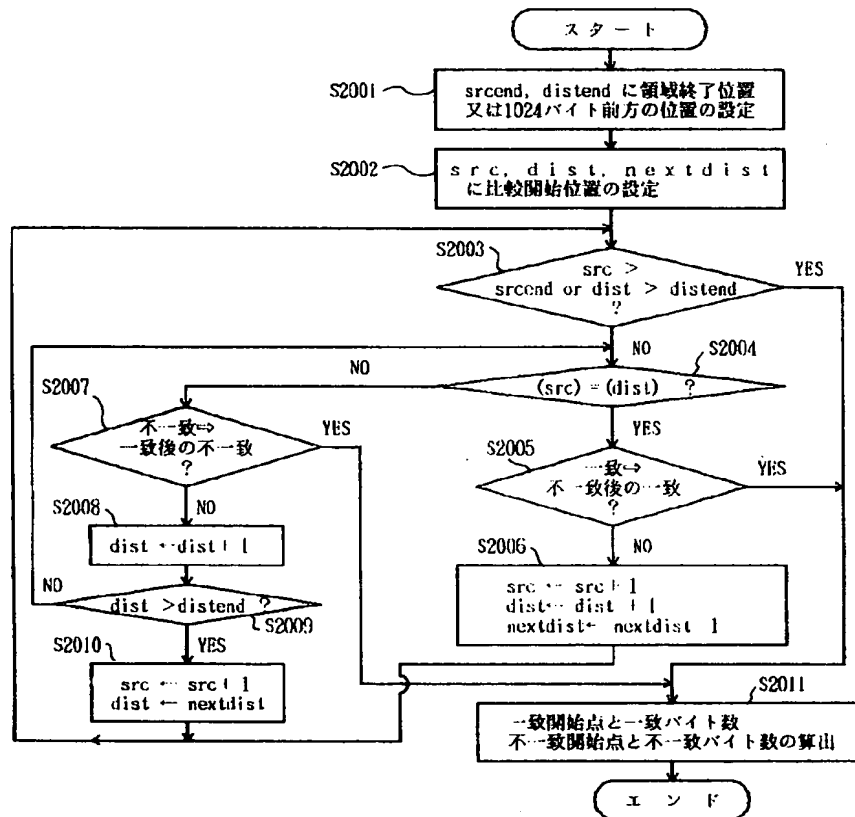
【図19】

差分抽出処理の動作フローチャート



【図20】

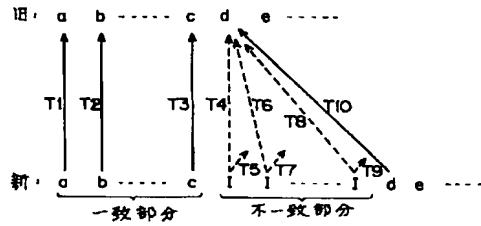
比較処理の動作フローチャート



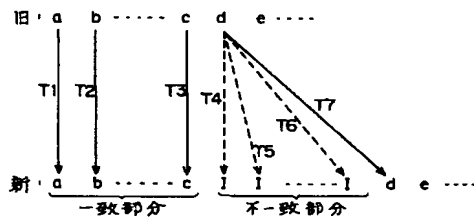
【図22】

新EXEファイルに挿入が行われた場合の差分抽出
処理の説明図

(a) 新EXEファイル → 旧EXEファイル



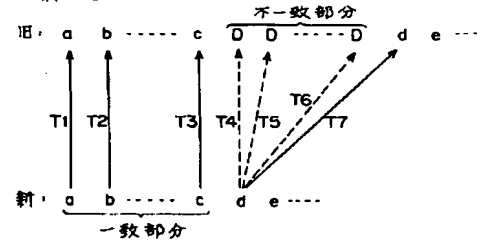
(b) 旧EXEファイル → 新EXEファイル



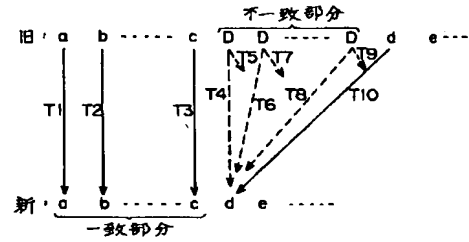
【図23】

新EXEファイルで削除が行われた場合の差分抽出
処理の説明図

(a) 新EXEファイル → 旧EXEファイル



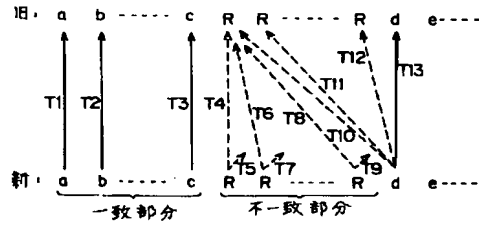
(b) 旧EXEファイル → 新EXEファイル



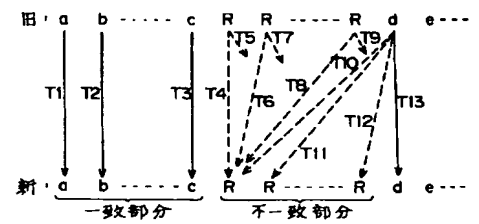
【図24】

新EXEファイルで置換が行われた場合の差分抽出処理の説明図

(a) 新EXEファイル → 旧EXEファイル



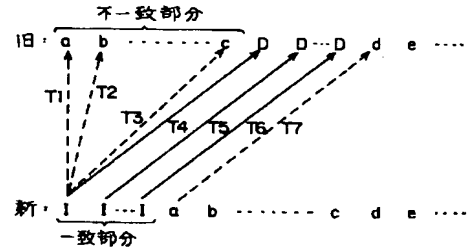
(b) 旧EXEファイル → 新EXEファイル



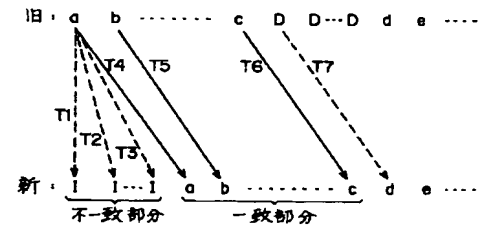
【図26】

新EXEファイルで特殊な挿入及び削除が同時に行われた場合の差分抽出処理の説明図

(a) 新EXEファイル → 旧EXEファイル

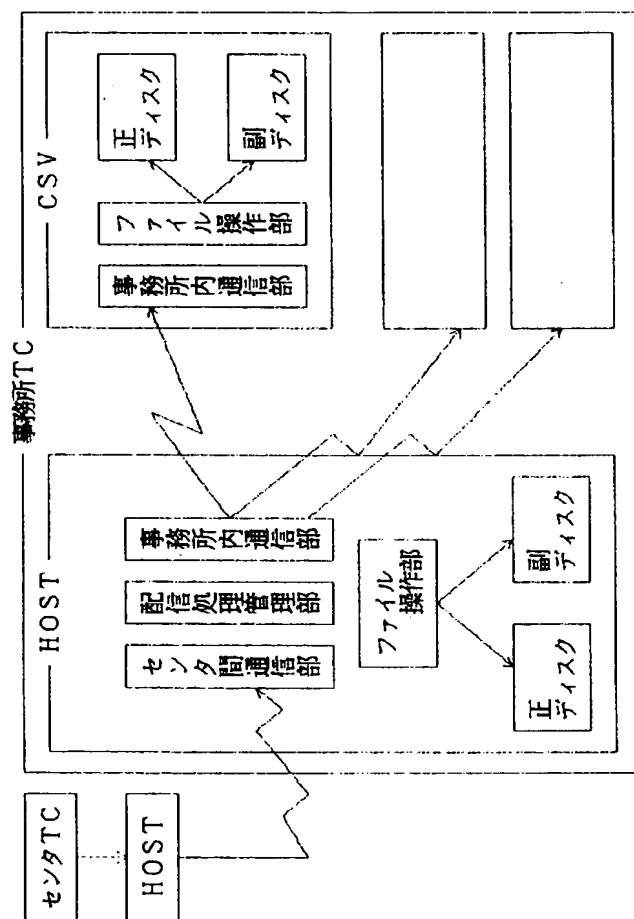


(b) 旧EXEファイル → 新EXEファイル



【图 29】

ファイル更新環境の説明図



**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☒ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☒ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.